Dynamic Control and Singularities of Rigid Bearing-Based Formations of Quadrotors

Julian Erskine

CENTRALE NANTES

THÈSE DE DOCTORAT DE

L'ÉCOLE CENTRALE DE NANTES

ÉCOLE DOCTORALE N° 601
Mathématiques et Sciences et Technologies de l'Information et de la Communication
Spécialité : *Automatique, Productique, et Robotique*

Par

Julian Benedict Czapalay ERSKINE

Dynamic Control and Singularities of Rigid Bearing-Based Formations of Quadrotors

Thèse présentée et soutenue à l'École Centrale de Nantes, le 03 décembre 2021
Unité de recherche : UMR 6004, Laboratoire des Sciences du Numerique de Nantes (LS2N)

Rapporteurs avant soutenance :

Antonio FRANCHI	Chargé de recherche HDR CNRS, University of Twente, Enschede, Pays-Bas
Guillaume ALLIBERT	Maître de conférences HDR, Université de Côte d'Azur, Sophia Antipolis

Composition du Jury :

Président :	Paolo ROBUFFO GIORDANO	Directeur de Recherche CNRS, INRIA Rennes
Examinatrice :	Ouiddad LABBANI-IGBIDA	Professeure des Universités, Université de Limoges
Dir. de thèse :	Isabelle FANTONI	Directrice de Recherche CNRS, École Centrale de Nantes
Co-encadrant :	Abdelhamid CHRIETTE	Maître de Conférences, École Centrale de Nantes

Invité:

Sébastien BRIOT	Chargé de Recherche HDR CNRS, École Centrale de Nantes

ACKNOWLEDGEMENTS

THIS work was made possible through funding from the *région Pays de la Loire*, the *École Centrale de Nantes*, the *Centre National de la Recherche Scientifique (France)*, and the *Natural Sciences and Engineering Research Council of Canada*.

First of all, I would like to thank my thesis director Isabelle Fantoni for taking me on as a student, organising funding, and guiding my scientific research. I would also like to thank my co-supervisor Abdelhamid Chriette for his support throughout my master and PhD theses. Sébastien Briot was another critical contributor to this thesis, and provided many valuable ideas on the analysis of formation singularities. Along with Stéphane Caro who worked with me on a project on aerial cable robots, their mentorship in conducting research and writing papers was of enormous value. I would also like to thank all the members of my thesis jury for their time and effort spent reviewing my work, and particularly to the rapporteurs Guillaume Allibert and Antonio Franchi for their detailed and constructive reports.

I must also acknowledge the many members of the LS2N ARMEN team (and others) who played a role in the development of this thesis. Franco and Olivier provided great advice on programming, while the technical support of Rafael helped to get my experiments up and running. Thanks as well to Damien for many interesting technical discussions, and to Sébastien Rouquet who helped me get my first drone off the ground. Collaborations with Shiyu, Minh-Quan, Zhen, Nicola, and Kojo all contributed to enrich my PhD experience. Finally my friends and colleagues, Guillaume, Valentin, Maël, Sam, David, and Zhongmou who help to push me through the difficult moments of this PhD.

Finally, I would never have been able to complete my thesis without the support of my friends and family. Ludivine, I would never have been able to do this without you. You were always supportive when I needed it, and would help me to plan a path forward when I couldn't find one myself. Je me tien également a remercié Mimi et Pascal pour m'avoir integrer dans leurs famille ces trois derniers années. And last but not least, the love and encouragement of my mother Ava, my father Neil, my siblings Rachel and David, and all the rest of my family back in Canada pushed me to go as far as I could.

ABSTRACT

ACHIEVING cohesive behaviour from a group of individuals is one of the great challenges in modern robotics. This thesis attempts to tackle one type of group behaviour, moving in formation, for underactuated multirotor unmanned aerial vehicles (UAVs) which have risen in popularity due to low price, high agility, and steady hovering. In particular, we treat the case of multirotors without access to absolute localization, and which detect one another by bearing measurements, which are extractable from onboard monocular cameras. With each UAV able to control its pose relative to its neighbours using local bearings, the entire fleet may be piloted as a single body, reducing the difficulty in directing (either manually or autonomously) potentially large numbers of UAVs. It is thought that in the future this may be of use for the efficient exploration of GPS/GNSS-denied areas, for robust redundant patrolling, and many other tasks. This thesis contributes to the state of the art of UAV bearing-based formations along two major axes: the development of dynamic formation controllers, and the analysis of singular embeddings of otherwise rigid formation structures.

Previous works on UAV bearing formations have resulted in controllers that are effective at low accelerations, considering UAVs as simple integrators and the visual feature evolution as a first-order process. These assumptions however ignore the non-linear dynamics of the UAVs and bearings, and therefore are not necessarily efficient for aggressive formation convergence and manoeuvring. In this thesis, we present two bearing formation control methods based on second order visual servoing (SOVS) and model predictive control (MPC) that take into account increasingly complex robot and interaction models. Factors such as numerical conditioning, actuation limits, local minima, and visual field-of-view are studied. Experiments show that these controllers can manoeuvre the desired formation at speeds exceeding 5 ms^{-1} and horizontal accelerations in excess of 10 ms^{-2} in a small indoor flying arena. We also demonstrate that the MPC method is well adapted to real-world scenarios, flying a wide formation through a narrow window using bearings extracted from in-flight images. Extensive simulations confirm that both the SOVS and MPC formation control

methods have potential to outperform existing bearing formation controllers in a wide range of scenarios (such as at high speeds, in sharp turns and with poorly estimated parameters), and a detailed analysis of their respective benefits and drawbacks are presented.

The second main contribution of this thesis is a detailed analysis of the degenerate configurations (or *singularities*) of rigid bearing formations. If a formation is rigid then the shape is uniquely defined by the local inter-robot measurements. Without rigidity, the desired shape of the formation cannot not be guaranteed, rigidity is thus crucial to the main function of formation control: piloting a group of UAVs as a single body. The rigidity of formations have been well studied from a combinatorial perspective using graph theory, however these methods only guarantee rigidity for *most* embeddings. For all formations there are special sets of embedding where singularities occur, resulting in additional degrees of freedom between two or more robots, potentially leading to collisions. Previously, singularities could only be identified for small formations, or numerically for a given formation with a known (and centralized) set of measurements. The "hidden robot" concept was adapted to bearing rigidity analysis prior to this thesis, allowing the UAV constraints and measurements to be expressed as a mechanism with physical kinematic constraints limiting the possible motions of the robots. This allows the application of tools such as screw theory to be applied to the study of rigidity, however it is found to be impractical beyond small formations. This thesis presents (to the best of our knowledge) the first general and systematic geometric analysis of the singularities of bearing formations, implementing a set-based contraction of constraints derived from the hidden robot model. This allows many singularities of relatively large formations (20 or more robots) to be identified without any ad-hoc kinematic analysis. We furthermore demonstrate how a class of arbitrarily large formation graphs can be created for which all singularities are known, guaranteeing the rigidity of the formation for all configurations lying outside of a known set.

It is hoped that the work presented in this thesis will demonstrate the benefit of considering non-linear robot models in formation control algorithms, and will permit the operation of multi-robot formations with faster dynamics and guaranteed rigidity.

Keywords

Quadrotor, Bearing formation, Multi-robot system, Predictive control, Visual Servoing, Singularities, Graph rigidity

RESUMÉ

ATTEINDRE un comportement cohésif de la part d'un groupe d'individus est l'un des grands défis de la robotique moderne. Cette thèse tente d'aborder un type de comportement de groupe, le déplacement en formation, pour les drones multirotors sous-actionnés qui ont gagné en popularité en raison de leur faible prix, de leur grande agilité et de la stabilité de leur vol stationnaire. En particulier, nous traitons le cas de multirotors qui n'ont pas accès à la localisation absolue et qui se détectent mutuellement par des mesures de cap (ou "*bearing*" ce qui indique la direction relative à l'observateur), qui peuvent être extraites de caméras monoculaires embarquées. Chaque drone étant capable de contrôler sa pose par rapport à ses voisins à l'aide de mesures visuelles localles, la flotte entière peut être pilotée comme un seul corps, ce qui réduit la difficulté de diriger (manuellement ou de manière autonome) un nombre potentiellement important de drones. On pense qu'à l'avenir, cela pourrait être utile pour l'exploration efficace de zones dépourvues de GPS/GNSS, pour des patrouilles redondantes robustes et pour de nombreuses autres tâches. Cette thèse contribue à l'état de l'art des formations de drones basées sur des *bearings* selon deux axes majeurs : le développement de contrôleurs de formations dynamiques, et l'analyse des configurations singulières des structures de formations génériquement rigides.

Les travaux antérieurs sur les formations de drones basées sur des *bearings* ont abouti à des contrôleurs qui sont très efficaces à de faibles accélérations, en considérant les drones comme de simples intégrateurs et l'évolution des caractéristiques visuelles comme un processus du premier ordre. Ces hypothèses ignorent cependant la dynamique non linéaire des drones et des *bearings*, et ne sont donc pas nécessairement efficaces pour la convergence et les manœuvres agressives des formations. Dans cette thèse, nous présentons deux méthodes de contrôle de flottes de drones en formation basées sur l'asservissement visuel de second ordre (SOVS) et la commande prédictive (MPC) qui prennent en compte des modèles de robot et d'interaction de plus en plus complexes. Des facteurs tels que le conditionnement numérique, les limites d'actionnement, les minima locaux et le champ de vision sont étudiés. Des expériences montrent que ces contrôleurs peuvent manœuvrer la

formation souhaitée à des vitesses supérieures à 5 ms^{-1} et à des accélérations horizontales supérieures à 10 ms^{-2} dans une petite arène de vol intérieure. Nous démontrons également que la méthode MPC en particulier est bien adaptée aux scénarios du monde réel, en faisant voler une formation large à travers une fenêtre étroite à l'aide de repères extraits d'images en vol. Des simulations approfondies confirment que les méthodes de contrôle de formation SOVS et MPC ont le potentiel de surpasser les contrôleurs de formation basée sur *bearings* existants dans un large éventail de scénarios (à des vitesses élevées, dans des virages serrés et avec des paramètres mal estimés), et une analyse détaillée de leurs avantages et inconvénients respectifs est présentée.

La deuxième contribution principale de cette thèse est une analyse détaillée des configurations de dégénérations (ou *singularités*) des formations de multirotors qui sont rigides dans des configurations génériques. Si une formation est rigide, sa forme est définie de manière unique par les mesures locales inter-robots. Sans rigidité, la forme désirée de la formation ne peut être garantie. La rigidité est donc cruciale pour la fonction principale du contrôle de formation : piloter un groupe de drones comme un seul corps. La rigidité des formations a bien été étudiée d'un point de vue combinatoire en utilisant la théorie des graphes, cependant ces méthodes ne garantissent la rigidité que pour *presque toutes* les configurations. Pour toutes les formations, il existe des ensembles spéciaux de configurations singulières, ce qui entraînent des degrés de liberté supplémentaires entre plusieurs robots, pouvant mener à des collisions. Auparavant, les singularités ne pouvaient être identifiées que pour de petites formations, ou numériquement pour une formation donnée avec un ensemble connu (et centralisé) de mesures. Le concept de "robot caché" a été adapté à l'analyse de la rigidité des graphes de *bearings* avant cette thèse, permettant aux contraintes des drones et aux mesures d'être exprimées comme un mécanisme avec des contraintes cinématiques physiques limitant les mouvements possibles des robots. Cela permet l'application d'outils tels que la théorie des torseurs à l'étude de la rigidité, cependant il s'avère être peu pratique au-delà des petites formations. Cette thèse présente (à notre connaissance) la première analyse géométrique générale et systématique des singularités des formations basée sur des *bearings*, en mettant en œuvre une contraction par ensembles de contraintes dérivées du modèle caché du robot. Ceci permet d'identifier de nombreuses singularités de formations relativement grandes (20 robots ou plus) sans aucune analyse cinématique ad-hoc. Nous démontrons en outre comment une classe de graphes de formation arbitrairement grands

peut être créée pour laquelle toutes les singularités sont connues, garantissant la rigidité de la formation pour toutes les configurations situées en dehors d'un ensemble connu.

Nous espérons que le travail présenté dans cette thèse démontrera l'avantage de considérer des modèles de robots non linéaires dans les algorithmes de contrôle de formation, et permettra l'opération de formations multi-robots avec une dynamique plus rapide et une rigidité garantie.

Mots Clés

Quadrirotor, Commande de flottes, Systeme multi-robots, Commande prédictive, Asservissement visuel, Singularitiés, Rigidité de graphes

CONTENTS

LIST OF FIGURES

LIST OF TABLES

List of Media

Chapter	Description and link	QR code
Chapter 4	Demonstration of the effect of different centrifugal force compensation strategies for the SOVS controller https://box.ec-nantes.fr/index.php/s/WxPKXxeJ3MweLNC	
Chapter 5	Experiments for the basic MPC formation controller https://box.ec-nantes.fr/index.php/s/FfrWWrwLaTfHoF5	
Chapter 5	Simulations of the FOV constrained MPC showing local minima with and without terminal costs https://box.ec-nantes.fr/index.php/s/4b5pXJ4ZxCMkiTm	
Chapter 5	Experiments testing MPC with onboard visual detection and FOV constraints https://box.ec-nantes.fr/index.php/s/KxBHEJLatQgDJdG	
Chapter 5	Experiments testing MPC with onboard visual detection and obstacle constraints https://box.ec-nantes.fr/index.php/s/Z3woYNJfD6XEyLG	
Chapter 6	Simulations of SOVS and MPC formation controllers following a lemniscate trajectory https://box.ec-nantes.fr/index.php/s/o2KGZZncPGdW6mk	
Chapter 6	Simulations of SOVS and MPC formation controllers following a square trajectory https://box.ec-nantes.fr/index.php/s/fTQbXLNEqSBWwTj	
Chapter 7	Demonstration of the effect of singularities on a simulated formation using MPC https://box.ec-nantes.fr/index.php/s/AY3EoLqZLRmHCFB	
Chapter C	Demonstration of mechanical singularities for an understanding of screw theory analysis https://box.ec-nantes.fr/index.php/s/eLpD5iQLTxR9psE	

GLOSSARY

Abbreviations

EKF	Extended Kalman Filter
COM	Center of Mass
DOF	Degrees of Freedom
FCU	Flight Control Unit
GPS	Global Positioning System
IMU	Inertial Measurement Unit
MOCAP	Motion Capture System
MPC	Model Predictive Control
P	Proportional
PD	Proportional Derivative
PID	Proportional Integral Derivative
ROS	Robot Operating System
RTI	Real Time Iteration
SOVS	Second Order Visual Servoing
SQP	Sequential Quadratic Programming
SVD	Singular Value Decomposition
UAV	Unmanned Aerial Vehicle
VTOL	Vertical Takeoff and Landing

Symbols

This section contains a list of symbols used in the thesis. We use the convention:

- Lower case symbols "x" indicates a scale value
- Bold lower case symbols "$\mathbf{x}$" represents a column vector
- Bold upper case symbols "$\mathbf{X}$" represents a non-vector matrix
- Italicized upper case symbol "X" represent physical points
- Fancy symbols such as "$\mathbb{X}$", "$\mathcal{X}$", and "$\mathfrak{X}$" represent non-numerical information such as mathematical structure, sets, manifolds, robots, etc...

Manifolds

$\mathbb{R}^1$	1-dimensional cartesian space
$\mathbb{R}^2$	2-dimensional cartesian space
$\mathbb{R}^3$	3-dimensional cartesian space
$\mathbb{S}^1$	1-dimensional rotational space

$\mathbb{S}^2$	The manifold spanning the surface of a unit sphere
$SO(3)$	The special orthonormal group spanned by three orthogonal rotations
$SE(2)$	The manifold spanned by $\mathbb{R}^2 \times \mathbb{S}^1$
$\mathbb{R}^3 \times \mathbb{S}^1$	The manifold spanned by 3D translational space and a rotation around a vertical axis
$SE(3)$	The manifold spanned by $\mathbb{R}^3 \times SO(3)$

General Notation

$\mathfrak{F}_0$	Inertial reference frame
$\boldsymbol{\mathcal{I}}_n$	Identify matrix of dimension n
$\mathbf{e}_n$	A vector of zeros (usually of dimension 3) except for the n^{th} element which is 1
$\mathbf{g}$	Gravity vector of [0 0 -9.81]$^{\mathrm{T}}$ ms^{-2}

Multirotor Notation

$\mathfrak{F}_i$	Body-fixed frame of quadrotor i using East-North-Up (ENU) convention
m_i	Mass of quadrotor i
$\mathbf{p}_i$	Position of quadrotor i
$\mathbf{v}_i$	Velocity of quadrotor i
$\mathbf{a}_i$	Acceleration of quadrotor i
$\mathbf{q}_i$	Attitude of quadrotor i as a unit quaternion
$\mathbf{R}_i$	Attitude of quadrotor i as a rotation matrix
$\phi_i\ \theta_i\ \psi_i$	Attitude of quadrotor i in roll-pitch-yaw euler angles
$\boldsymbol{\omega}_i$	Body-frame angular velocity of quadrotor i
f_i	Thrust magnitude of quadrotor i
f_p	Thrust magnitude of propeller p
$\mathbf{f}_i$	Thrust vector of quadrotor i as $\mathbf{f}_i = f_i\mathbf{R}_i\mathbf{e}_3$ in $\mathfrak{F}_0$
$\boldsymbol{\tau}_i$	Moment vector exerted on quadrotor i by it's differential propeller actuation
$\boldsymbol{\Omega}_i$	Vector of propeller angular velocities of quadrotor i

Formation Modelling Notation

$\mathcal{A}_i$	The i^{th} agent (or robot) in the formation
$\mathcal{V}_i$	The graph vertex representing $\mathcal{A}_i$
$\boldsymbol{\mathcal{V}}$	The set of all vertices in a graph
$\mathcal{E}_{ij}$	A directed edge from $\mathcal{V}_i$ to $\mathcal{V}_j$
$\boldsymbol{\mathcal{E}}$	The set of all edges in a graph
$\mathcal{G}$	Graph $\mathcal{G}(\boldsymbol{\mathcal{V}}, \boldsymbol{\mathcal{E}})$ representing the robots and their interactions in the formation
$\mathcal{N}_i$	Neighbourhood of graph around vertex $\mathcal{V}_i$
$\mathbf{q}_i$	The embedding vector of $\mathcal{A}_i$ consisting of its flat outputs $\mathbf{q}_i = [\mathbf{p}_i^T\ \psi]^T$
$\mathbf{q}$	The stacked embedding vectors $\mathbf{q} = [\mathbf{q}_1^T\ \ldots \mathbf{q}_{\lvert\boldsymbol{\mathcal{V}}\rvert}^T]^T$
$\mathcal{F}$	The framework $\mathcal{F}(\mathcal{G}, \mathbf{q})$ of a formation consisting of a graph and an embedding vector

$\boldsymbol{\beta}_{ij}$	The bearing vector of $\mathcal{A}_j$ as observed by $\mathcal{A}_j$ in $\mathfrak{F}_i$
$\boldsymbol{\beta}$	The stacked vector of all all measured bearings $\boldsymbol{\beta} = [\boldsymbol{\beta}_{1j}^T \dots \boldsymbol{\beta}_{\lvert\boldsymbol{\mathcal{E}}\rvert k}^T]^T$
$\mathbf{B}$	The bearing rigidity matrix $\mathbf{B} = \frac{\partial \boldsymbol{\beta}}{\partial \mathbf{q}}$
λ_R	The bearing rigidity eigenvalue (the sixth smallest eigenvalue of $\mathbf{B}^T\mathbf{B}$)

Formation Control Notation

$\mathbf{u}_i$	The vector of control variables for the formation controller
$\boldsymbol{\beta}_i$	The vector of all bearings $\boldsymbol{\beta}_i = [\boldsymbol{\beta}_{ij}^T \ \dots \ \boldsymbol{\beta}_{in}^T]^T$ measured by $\mathcal{A}_i$
d_{ij}	The (generally unknown) distance $d_{ij} = \lvert\lvert\mathbf{p}_j - \mathbf{p}_i\rvert\rvert$ between $\mathcal{A}_i$ and $\mathcal{A}_j$
$\mathfrak{M}$	A vector of weights $[\mathbf{v}_{\mathcal{F}}^T \ \dot{\psi}_{\mathcal{F}} \ \dot{s}_{\mathcal{F}}]^T$ to be projected in $\ker(\mathbf{B})$
$\mathbf{c}_{\mathcal{F}}$	The center of mass of the formation
$\mathbf{P}_{ij}$	The orthogonal projection matrix with respect to $\boldsymbol{\beta}_{ij}$
$\mathbf{L}_{ij}$	The bearing interaction matrix on the $\mathbb{R}^3 \times \mathbb{S}^1$ manifold
$\mathbf{h}_{ij\lvert k}$	The hessian vector of the bearing interaction model with respect to the motion of $\mathcal{A}_k$
T_p	The prediction horizon
Δt	The prediction time step
$\mathbf{x}_d$	The desired values of some system property $\mathbf{x}$
$\mathbf{e}_x$	Vector of errors resulting from $\mathbf{e}_x = \mathbf{x}^d - \mathbf{x}$
$\boldsymbol{\epsilon}$	Vector of slack variables $\boldsymbol{\epsilon} = [\epsilon_1 \dots \epsilon_n]^T$
$\mathcal{O}$	An objective (or cost) function to be minimized in an optimization problem
$\mathcal{C}$	A set of constraints to be respected in an optimization problem

Formation Singularity Notation

$\mathbf{p}_{ij}$	A straight line in $\mathbb{R}^3$ passing through $\mathcal{A}_i$ and $\mathcal{A}_j$
$\mathbf{p}_{ij}^{\perp}$	A straight line orthogonal to $\mathbf{p}_{ij}$
$\$$	General unit screw $\$(\mathbf{s}, \mathbf{p}, \lambda)$ along axis $\mathbf{s} \in \mathbb{R}^3$ at position $\mathbf{p} \in \mathbb{R}^3$ and pitch λ
$\mathbb{v}^{\mathrm{t}}$	A infinite-pitch velocity screw (twist) representing pure translational motion
$\mathbb{v}^{\mathrm{r}}$	An zero-pitch velocity screw (twist) representing pure rotational motion
$\mathcal{T}$	A set of twists
$\mathbb{w}^{\mathrm{f}}$	A zero-pitch effort screw (twist) representing a pure force
$\mathbb{w}^{\mathrm{m}}$	A infinite-pitch effort screw (wrench) representing pure moment
$\mathcal{W}_m$	A set of wrenches
$\mathcal{S}^L$	Set of local singularities
$\mathcal{S}^F$	Set of subformation singularities
$\mathbb{O}$	An instance of a graph edge $\mathcal{E}_{ij}$ leaving $\mathcal{V}_i$
$\mathbb{I}$	An instance of a graph edge $\mathcal{E}_{ji}$ entering $\mathcal{V}_i$
$\mathbb{B}$	A dual instance of graph edges $\mathcal{E}_{ij}$ and $\mathcal{E}_{ji}$

Operators

$\dot{x}$	The time derivative of x
x_{ij}	The element locate at the i^{th} row and j^{th} column of matrix $\mathbf{X}$
$x_{x,i}$	The x component of vector $\mathbf{x}_i$
sign(x)	A function that returns 1 if $x \geq 0$ and -1 if $x < 0$
$[\mathbf{x}]_\times$	The skew symmetric matrix (or cross product matrix) such that $[\mathbf{x}_1]_\times \mathbf{x}_2 = \mathbf{x}_1 \times \mathbf{x}_2$
$\mathbf{x}_1 \times \mathbf{x}_2$	The cross product operation
$\lvert\mathbf{x}\rvert$	The cardinality (total number of elements) of a set or vector
$\lvert\lvert\mathbf{x}\rvert\rvert$	The 2-norm of vector $\mathbf{x}$
$\mathbf{x}^T$	The transpose of vector $\mathbf{x}$
$\mathfrak{q}_1 \otimes \mathfrak{q}_2$	The quaternion product operator
$\$_i \circ \$_j$	Reciprocal product of screws i, j
$\mathbf{X}^{-1}$	The matrix inverse of $\mathbf{X}$ (assuming that $\mathbf{X}$ is a square matrix)
$\mathbf{X}^\dagger$	The Moore-Penrose pseudo-inverse of $\mathbf{X}$, as $\mathbf{X}^\dagger = \left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T$
$\ker(\mathbf{X})$	The kernel (or a basis spanning the nullspace) of $\mathbf{X}$
rank$(\mathbf{X})$	The rank of $\mathbf{X}$
$\mathbf{x} \in \mathbb{X}$	$\mathbf{x}$ is contained within set $\mathbb{X}$
$\mathbb{X}_1 \cup \mathbb{X}_2$	The union of sets $\mathbb{X}_1$ and $\mathbb{X}_2$
$\mathbb{X}_1 \cap \mathbb{X}_2$	The intersection of sets $\mathbb{X}_1$ and $\mathbb{X}_2$
$\mathbb{X}_1 \subset \mathbb{X}_2$	$\mathbb{X}_1$ is a subset of $\mathbb{X}_2$

PART I

Introduction and Background Material

Abstract

This first part of the manuscript provides the necessary context to understand the scope of this thesis, and the state of current work in the field. Chapter 1 details the structure of the manuscript, and how our work fits into the overarching and vast field of multi-robot systems. The second chapter is dedicated to the understanding on a technological and theoretical level of unmanned aerial vehicles. The third and final chapter in this part discussed multi-robot systems including the necessary tools such as graph theory that is current in the domain.

List of Chapters

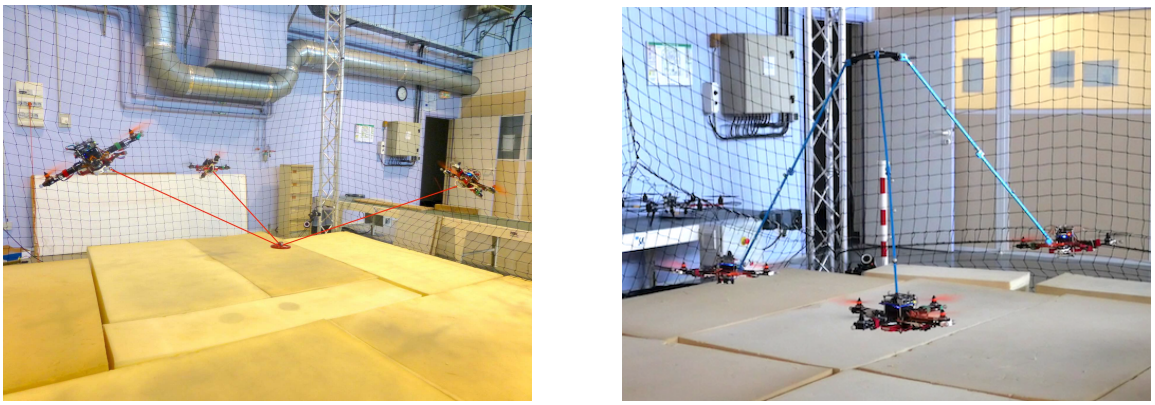
CHAPTER 1

OVERVIEW OF THESIS

THE opening chapter of this dissertation provides the reader with an understanding of the work performed in the scope of this doctoral thesis. The context of the research is presented in the introduction section, followed by a discussion on contributions made during this thesis. Finally, an outline of the contents and structure of this document is provided.

1.1 Background and Motivation

The 21st century has seen robots leave their strictly controlled industrial environments, operating with increasing independence and versatility. While the word "robot" may still evoke the image of an anthropomorphic arm or even a full humanoid robot, the domain has expanded to include (among many others) self-driving cars, aerial vehicles [1], and other mobile autonomous robots. Advancements in these fields have largely been driven by developments in battery and computing technology, which has been steadily trending to lighter weight and high power. As the autonomy of mobile robots allows them to be deployed in more remote locations, new challenges must be overcome. These hitherto inaccessible environments (such as inside buildings, tunnel networks, or under tree-cover) may impose constraints such as the unavailability of absolute localisation, interrupted communications,



(a) An aerial cable-suspended parallel robot (b) A flying rigid parallel robot

Figure 1.1 – Aerial robotics platforms at LS2N

and difficult accessibility. Multi-agent systems are seen by many researchers as an attractive way of overcoming these constraints, as using many small robots overcomes the accessibility constraint, and can act as a distributed sensor network permitting mutual localisation. Furthermore, a wide range of tasks benefiting from redundancy, increased coverage, and variable scaleability may benefit from being performed by a group of smaller autonomous robots.

This thesis focuses specifically on aerial robots, which have become popular for both consumer and commercial uses. The Laboratoire des Sciences du Numerique de Nantes (LS2N) began work on multi-robot aerial systems by developing parallel flying robots for aerial manipulation and inspection (see Fig. 1.1) [2]–[5]. In 2018, the AtlanSTIC2020 project RAPID[1] was proposed to extend the lab's existing fields of expertise in aerial robotics, visual control, and parallel robotics to the control of drone fleets. This was inspired by the recent theses from UTC Heudiasyc in Compiegne [6] and INRIA Rennes [7], with the idea that there may be strong links between formations of quadrotors and existing parallel mechanism modelling and control. The objectives of project RAPID include

1. Control of drone fleets: High-speed formation control has been implemented in open areas and with a high degree of available information. When there is less information available (i.e. robots are restricted to using onboard measurements) methods such as rigidity-based control are used, but often are limited to simple linear robot models using single or double integrators. This projects intends to extend

1. The AtlanSTIC2020 program supports scientific and technological research in the Pays de la Loire region of France. https://atlanstic2020.fr. RAPID is an acronym for "Formation de **R**obots **A**ériens hétérogènes pour l'ex**P**lorat**I**on avec des **D**ynamiques variables"

decentralized rigidity-based controllers to include more complicated non-linearities for more aggressive formation flight in GPS-denied environments.

2. State estimation in drone fleets: Formation control requires an estimation of relative robot states through mutual localisation algorithms. An objective was to study inter-robot localization, including the development of new algorithms for localization, and also to study the singular conditions for which mutual localization is no longer possible.

3. Data fusion with dynamically variable sensors : New sensors such a event-based cameras [8] have begun to change how robots can perceive their environment, capturing changes in scene rather than a change in scene itself. It is thought that the integration of these cameras in to quadrotor formations could give rise to new applications but also new challenges in control and state estimation.

As expected, this thesis has not been able to deal with all objectives, however significant work has been made on some, as summarized in the following section and detailed throughout this manuscript.

1.2 Contributions

This section outlines the contributions made during the three years of this thesis, from september 2018 to september 2021. First we describe how the broader objectives of the project are narrowed down, and then we summarize the novel contributions of made in the field of formation control. We also briefly mention the scientific contributions made during the thesis that were not performed in the scope of project RAPID, and thus are not covered in detail by this dissertation.

1.2.1 Contributions to Quadrotor Formation Control

The control of drone formations as described in the title of RAPID encompasses many possibilities. Early on, it was decided to restrict the scope of the work to make specific contributions to drone formation control that fit most with my interests and background, as well as the research expertise of the ARMEN group at LS2N on sensor-based control and singularity analysis. We therefore chose to focus specifically on bearing formation control for

this thesis, as it may be performed using onboard vision and because the singularities of these formation had been the subject of an initial study within our team and it was thought that there was ample opportunity to make scientific contributions. With regards to the control of formations, much of the existing work on bearing formation control deals with using rigidity to develop global proofs of convergence which can be decentralized and applied on a local level. These methods however make use of simplified robot models, and we believed that assuming certain basic global properties of the formation could be satisfied, the control of formations can be most improved by considering more complete robot and local interactions.

This thesis is therefore able to situate itself in the midst of a crowded research field by

1. **Improving dynamic bearing formation control**: We consider more complicated interaction models and robot kinematics than are typically used for bearing formation control by using second-order visual servoing (SOVS) and model predictive control (MPC). This lead us to achieve formation controllers with fast transient formation convergence rate, and horizontal velocities of 5 ms^{-1} when steering the formation in experiment. This is significantly faster than any other decentralized onboard sensor-based formation controller that we have found in literature. We furthermore show that with MPC, the formation control can be augmented with practical constraints on the actuation, the camera field of view, and obstacle avoidance with little issue.

2. **The study of bearing formation singularities**: Bearing formations have long since been known to have singularities (configurations where the formation is not fully controllable). Previous work has demonstrated that this prevents the convergence of gradient-based controllers and estimators, but we that in practice the problem of singularities is the inability to guarantee a unique shape of the formation. By using kinematic analysis tools and by considering constraint-based sets, we are able to extend the formal analysis of bearing formation singularities from formations of 3-4 UAVs to formations of more than 30 UAVs. While the method is not guaranteed to find all singularities in all formations, we show how we can construct specific classes of formations (with an arbitrarily large number of robots) such that we are guaranteed to know all singular configurations.

A comprehensive list of all my works published during my PhD are presented in table 1.1. Only the items directly related to the PhD topic are presented in this dissertation (those associated with chapter numbers in table 1.1).

Table 1.1 – A list of the material published or currently under review for publication during the course of this thesis. The list is ordered from most recent to least recent.

Type	Citation	Chapters
Conference	Liu, S., **Erskine, J.**, Chriette, A., Fantoni, I. *"Decentralized Control and Teleoperation of a Multi-UAV Parallel Robot using Intrinsic Measurements"*, IEEE International Conference on Intelligent Robots and Systems IROS, Sept. 2021	None
Conference	**Erskine, J.**, Balderas-Hill, R., Fantoni, I., Chriette, A. *"Model Predictive Control for Dynamic Quadrotor Bearing Formations"*. IEEE International Conference of Robotics and Automation ICRA, May 2021	5
Journal	Li, Z., **Erskine, J.**, Caro, S., Chriette, A. *"Design and Control of Variable Aerial Cable-Towed Systems"*, IEEE Robotics and Automation Letters, Jan. 2020 (Also presented at ICRA 2020)	None
Journal	Six, D., Briot, S., **Erskine, J.**, Chriette, A. *"Identification of the Propeller Coefficients and Dynamic Parameters of a Hovering Quadrotor from Flight Data"*. IEEE Robotics and Automation Letters, Jan. 2020 (Also presented at ICRA 2020)	None
Conference	**Erskine, J.**, Chriette, A., Caro, S. *"Control and Configuration Planning of an Aerial Cable-Towed System"*. IEEE International Conference on Robotics and Automation ICRA, May. 2019	None
Journal	**Erskine, J.**, Chriette, A., Caro, S. *"Wrench Analysis of Cable-Suspended Parallel Robots Actuated by Quadrotors UAVs"*. ASME Journal of Mechanisms and Robotics, Feb. 2019	None
In Submission	**Erskine, J.**, Briot, S., Fantoni, I., Chriette, A. *"Singularities of Bearing Graph Rigidity with Application to Quadrotor Bearing Formations"*.	8
In Preparation	**Erskine, J.**, Fantoni, I., Chriette, A. *"Vision-Based Dynamic Quadrotor Formation Control using Constrained Model Predictive Control"*.	5-6

1.2.2 Other Contributions

During the span of my PhD thesis from September 2018 up until the submission of this manuscript, I participated in several other projects related to aerial robotics. Publications related to these projects (works in table 1.1 without a chapter number) are not detailed in this manuscript, but are briefly outlined here:

1.2.2.1 Aerial Cable-Suspended Parallel Robots

This work began during my second year master thesis, for which I was task with extending existing wrench analysis methods of cable-driven parallel robots to cable-suspended payloads carried by multiple quadrotors. I was also given the objective of designing, building, and programming a working prototype to validate the aforementioned analysis. While I succeeded in developping a static wrench analysis method [9] and a working prototype during the master thesis, there was insufficient time in the master thesis to fully valorize the work accomplished. I therefore devoted several months early in my PhD to extend the wrench analysis method to a dynamic system [4] and to the experimental validation of the prototype [5] (and in the process developping the UAVs which I would go on to use for formation control). To continue this work, during the first year of my thesis I co-supervised a new master student extending our original work to include quadrotors with embedded winches [10]. My involvement with the project ended in 2020 to focus on the core subject of this dissertation.

1.2.2.2 The Flying Parallel Robot

The flying parallel robot (FPR) consists of multiple quadrotors joined to an end effector by rigid links and passive joints. It was the first practical works on multi-UAV aerial manipulation, developed as the subject of the PhD thesis of D. Six (2015-2018) [11]. Different kinematic architectures were proposed and a working prototype was developed based off the quadrotors built for the aerial cable-suspended robot. When a new PhD student, S. Liu took over the FPR project, I proposed a collaborative work to overcome the limits of the previous controller of the FPR by applying concepts from decentralized formation control studied during my thesis.

Because the original FPR made used of a centralized control requiring high-precision positioning of the quadrotors and platform in an inertial frame. This implies the necessary

use of motion capture systems, imposing an impractical constraint of real-world applications. What we proposed was to place embedded cameras on each quadrotor, such that each quadrotor is able to estimate the pose of the platform in the quadrotor's local frame using visual pose estimation methods. Two decentralized controllers, one with and one without communication between the quadrotors are proposed and tested with a human pilot performing eye-to-hand teleoperation of the platform velocity in the platform's local frame. This work has been presented at IROS 2021 [12].

1.2.2.3 The Identification of Quadrotor Dynamic Parameters

This work extends industrial robot dynamic parameter identification techniques to multirotor UAVs. These UAVs typically have 12 parameters near hovering conditions (i.e. when the apparent wind speed is low); the mass, the position of the center of gravity ($\mathbb{R}^3$), the moments of inertia ($\mathbb{R}^6$), and the propeller lift and drag propeller coefficients. Generally the center of gravity is assumed to be at the centroid of the UAV, the inertias are estimated by CAD, and the propeller coefficients are measured by static test benches. We proposed a method to identify these parameters using a single flight with a known mass, recording the measured accelerometer data and the desired motor speeds [13]. My role in this work was less than in others, as the ideas were mostly developed by the first two co-authors prior to my arrival. I was responsible for the experiment procedure, designing components, piloting the quadrotor, and gathering data. I also contributed to the writing of the experiments section and the generalization of the UAV modelling in the paper.

1.3 Thesis Structure

This thesis is divided into four main parts. The first and last parts deal primarily with contextualizing the work done in this thesis and the presentation of background knowledge. The second and third parts present the scientific developments of this thesis, dealing primarily with the development and testing of controllers (part 2), and an analysis of singularities in the rigidity of formation structures (part 3). Finally the last part of this manuscript contains the conclusions, references, and other supporting material such as the appendices.

1.3.1 Part 1 - Preliminary Content

The first part treats the background and context of this thesis, and contains prerequisite and general state of the art. It is divided into three chapters:

- **Chapter 1** introduces the thesis, presenting the context for the work, along with the objectives and accomplishments. It also serves to outline the structure of this manuscript.

- **Chapter 2** presents an introduction to UAVs, starting with a discussion on their technological development and different types. A detailed model is presented for multirotor-type UAVs which are used in this thesis. One of the many existing control and estimation schemes for quadrotors is presented, along with details on common embedded sensors.

- **Chapter 3** provides a discussion on the existing and potential uses of multi-robot systems, and then presents the different architectures including the important notion of decentralization. Notions of graph theory are presented along with existing bearing-based formation control algorithms.

1.3.2 Part 2 - Contributions to Dynamic Formation Control

The second part deals with our original work on the control of quadrotor formations. Two controllers have been developed and tested and are presented individually in the first two chapters, and are compared in the final chapter:

- **Chapter 4** presents an approach to bearing formation control adapter from second-order visual servoing (SOVS). A background into second-order visual servoing applied to robotic systems is presented and then the model is adapted to quadrotor formation control. Simple simulations are used to demonstrate issues and with this control law, including numerical singularities, maneuvering the formation, and accounting for actuator saturation, and show that effective solutions are found. Real-time experiments confirm that the developed method is indeed effective.

- **Chapter 5** presents the second type of bearing formation controller developed in this thesis, this time based on model predictive control (MPC). A brief state of the art on MPC is provided, and then we formulate the optimization problem for our

controller. Simulations show the reliability of this method and the link between the prediction horizon and formation scale. Experiments demonstrate the effectiveness of this controller for speeds of 5ms^{-1}, in the presence of perturbations and model errors, and the implementation and effects of constraints including sensor fields-of-view and environmental obstacle avoidance.

- **Chapter 6** analyses the performance of the MPC and SOVS controllers compared to a state-of-the-art rigidity controller using a single integrator robot model. Extensive simulations demonstrate the performance of the developed controllers against an existing controller using single integrator robot kinematics. Tests include high speed trajectories, fast cornering, sensitivity to sensor noise and parameter uncertainties, and execution time.

1.3.3 Part 3 - Contributions to Bearing Formation Singularities

The third part of this thesis details the work on the singularities of quadrotor bearing formations, including a categorization of singularities, and an investigation into the behaviour of formations at or near singular configurations.

- **Chapter 7** is a chapter presenting primarily state of the art knowledge on graph rigidity singularities. First the nature of rigidity is mathematically and conceptually discussed, and then the hidden robot representation of bearing formations is presented along with notions and examples of screw theory for the later analysis of formation singularities.

- **Chapter 8** contains the novel contributions of this thesis towards the analysis of singular geometries in rigid formations. The complex problem of analysing large formations is decomposed into the simple analysis of substructures within the formation. We then categorize all the singularities for formations which are assumed to satisfy several assumptions. Case studies are used to present the application of this decomposed analysis to a complex formation. While the assumptions we made prevent all possible formations from being fully analysed, we show that it is possible to design formations of any size for which our analysis method is guarenteed to fully characterize all singular geometries.

1.3.4 Part 4 - Conclusions and Supplementary Material

The final part of the thesis presents the conclusions of the work performed and an outlook on the future of quadrotor formations. In also contains appendices with details such as experimental setups that are not included in other chapters.

- **Chapter 9** presents the conclusions of this thesis. The work and main contributions are summarized, and are then compared with the project objectives and with the state of the art research in the field. Prospectives on the continuation of this work are provided, along with key information to be taken away from this work.

- **Appendix A** presents the hardware, software and system architecture of the experiments performed within this thesis. The specifications and dynamic identification of the UAVs are detailed.

- **Appendix B** presents details on the extraction of bearings from onboard cameras. We discuss first how the bearings may be extracted from an onboard camera given it's position in the image, and we discuss the uncertainty due to detection error, delays, and camera offsets. The method used for detecting UAVs from in-flight images is presented, along with results showing the accuracy compared to the ground truth.

- **Appendix C** presents a example of the use of screw theory for the analysis of a simple mechanical structure, as a supplement to part 3. This is more of a tutorial for mechanism analysis in case the reader needs a practical example of the application of screw theory.

CHAPTER 2

INTRODUCTION TO UNMANNED AERIAL VEHICLES

FORMATION controllers may act independent of the type of agents, providing their low level controllers with reference trajectories to follow as best it can. However by using the agents' models, the formation controller may directly account for the motion constraints of the individual robots, potentially improving performance. This chapter therefore presents the modelling and control of multirotors that is used in the following chapters to develop new formation controllers.

First the modelling of quadrotor-like multirotors is presented, relating their actuation to their underactuated rigid-body motion. Then the typical control architecture of these multirotors with an outer translational and an inner attitude loop is shown. We then briefly discuss state estimation and trajectory generation of these UAVs, presenting the import

(a) The Montgolfiere balloon

(b) The Forlanini helicopter

(c) The Wright Brothers' plane

Figure 2.1 – Early flying machines [F1]–[F3]

concept of differential flatness and the flat outputs of the system.

2.1 A Brief Overview of Unmanned Aerial Vehicles

In this section, we first present a brief overview into the historical development of unmanned flying machines, discuss the different classes of UAV, and present a selection of current multirotors.

2.1.1 A History of Manned and Unmanned Flight

Humans have been fascinated by flight since antiquity (see the greek legend of Icarus), but it wasn't until the 15$^{\text{th}}$ century that a serious engineering-based (although ultimately unachieved) work on flying machines was attempted by Leonard DaVinci. Humans were only able to achieve rudimentary unmanned flying devices such as kites (several centuries B.C.) and small gas-filled balloons (1709) until the Montgolfier balloon's first manned flight (1783) near Paris (Fig. 2.1a). The problem with these flying machines was that they were not easily steerable and could operate either attached to a stationary tether, or at best move in the same direction as the prevailing wind of the moment [14].

The 19$^{\text{th}}$ century was marked by an exponential growth in mechanical power generation, first through the development of high-pressure steam engines and later due to the development of the internal combustion engine. The rapid increase of the power-to-weight ratio of these engines, along with advances in the science of aerodynamics (particularly influenced by George Cayley) began to make the prospect of heavier-than-air powered flight feasible [14]. The key properties of propulsion, lift, and control were identified, and powered flight was the objective of engineers around the world. This led to many firsts, including the first manned and powered flight by Felix Temple (1874), and the first unmanned multirotor flight by Enrico Forlanini in 1877 (see Fig. 2.1b), a 20 s flight by a steam-engine-powered,

biaxial helicopter [15].

Most readers will be generally familiar with the subsequent well document timeline of aviation, including the development of aerodynamically efficient gliders, the first sustained and controlled manned flight of the Wright brothers in 1903 (Fig. 2.1c) and the first manned helicopter flight (1907). Aviation became militarized and commercialized, with two primary designs emerging: fixed-wing planes and single-rotor helicopters powered first by internal combustion, and later by jet engines. The role of sensors, electronics, and computing became increasingly important, as the Canadian Avro Arrow (1958), and later the Concord (1969) began to make use of fly-by-wire systems where the manual control of the aircraft control surfaces by the pilot was partially given over to automated electrical controllers [16]. This allowed designers to reduce the aerodynamic stability of the aircraft as closed-loop control systems would constantly be making adjustments to stabilize the aircraft. This inherent instability allowed aircraft to be much more manoeuvrable. There have since been many impressive accomplishments in human flight including the SR-71 (speed > Mach 3), the Airbus A380 (850+ passengers) and even quasi-futuristic jetpacks [17] and flying cars [18]. It will be noted however that we have, since Forlanini's unmanned multirotor in 1877, left out the history of machines that fly themselves, unmanned aerial vehicles (UAVs).

Leaving aside such unmanned flying systems as rockets, we interest ourselves in UAVs capable of manoeuvrable, stable flight. While radio controlled UAVs (generally based on existing aeroplanes) with remote human guidance had existed since the 1930s, the development of specifically designed UAVs was accelerated during the cold war by the capture of an American U2 spy plane pilot in 1960 [19]. They continued however to be similar to traditional aircraft (albeit with sophisticated sensors), made lighter and faster due to the removal of human support system and with the advent of fly-by-wire [16]. The miniaturization of UAVs as we know them now, particularly unmanned multirotors, was made feasible through some critical technological advancements:

- The increased power density of electric batteries
- The decreased size of onboard computing
- The miniaturization of inertial sensors (particularly accelerometers and gyroscopes) and brushless DC motor controllers
- Advancements in GPS technology, with the weight of receivers decreasing from 16 kg

in 1991 to tens of grams now, and with sub-meter accurate absolute localisation becoming standard.

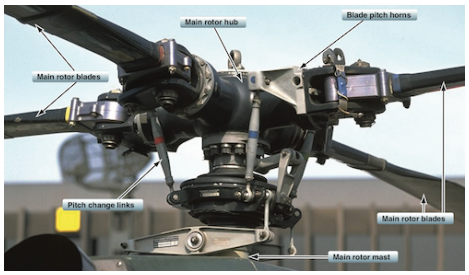
Several quadrotor-like designs had been tested in the 1920-30s but had been abandoned due to being infeasible due to the technological state of the time. Without high-speed sensing and computing the inherently unstable quadrotor was less feasible than the helicopter, with its central rotor located above its center of mass (COM). Furthermore, unlike helicopters, a large transmission was then required to move power from the central engine to the four distal propellers. With the previously mentioned advances, the quadrotor design became the subject of renewed interest in the academic field with several prototypes around the year 2000 [20]–[23]. From there, work on multi-rotor UAVs took off with both academic and commercial sectors developing better designs, navigation algorithms and flight control [24], [25].

In the past five years, the general trend in academic UAV related research has been towards task-based applications such as complex sensing or interaction tasks, instead of pure UAV control and design. Aerial manipulation has become a subject of immense interest, with numerous European projects [26], [27] among others. This includes the use of robotic arms mounted on UAVs [28]–[31] and the use of multiple UAVs for collaborative payload transportation [4], [32]–[34]. These advanced applications have encouraged the design of a new class of UAVs; omnidirectional multirotors. These multirotors are fully or even over-actuated, having increased manipulability over their classical quadrotor-style counterparts. Other work has included morphing or flexible UAV architectures such a [35], [36]. Nonetheless as we discuss subsequently, the quadrotor robots of the early 2000s remain a popular and versatile robotics platform.

2.1.2 A Comparison of Modern UAV Classes

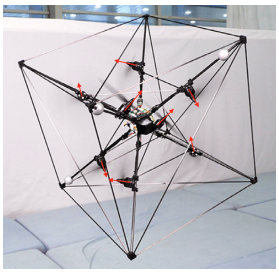
From the historical context provided, we can separate current UAVs into four main architectures as shown in Fig. 2.2: fixed-wing UAVs, helicopters, underactuated multirotors (such as quadrotors), and omnidirectional multi-rotors. While some other types exist such as ornithopters and morphing flying robots, these drones currently serve as mainly artistic or academic prototypes. A description of the benefits of the main types of UAVs are:

(a) Fixed-Wing [F4]



(b) Helicopter [F5]

(c) Quadrotor [F6]



(d) Omnidirectional [37]

Figure 2.2 – Different types of UAVs. The fixed-wing UAV is a drone by Zipline[1] for delivering medical supplies to remote communities. The helicopter figure shows the complexity of the rotor mechanism for changing the propeller pitch, necessary for steering the vehicle. A typical quadrotor is shown, which can be compared to the helicopter showing the relative mechanical simplicity. Finally an omnidirectional multirotor is shown.

1. **Fixed-wing UAVs** are generally adapted to long range missions, with industry variants having a range of hundreds of kilometres and military models having ranges greater than 25,000 km. They are also the fastest and the most payload capable UAVs. Fixed-wing civilian drone applications include medical supply delivery, search and rescue operations, and large-scale surveys of infrastructure such as power lines, train tracks and other geographically large features including mineral and agricultural exploitations. The primary downside of fixed-wing UAVs is that they need a runway or launch-assist device (such as a catapult) to take off, and to land. Furthermore, in operation they must maintain a minimum forward airspeed to avoid stalling and crashing. This results in a large turning radius, and an inability to hover at a fixed position.

2. **Helicopter UAVs** are often used when vertical takeoff and landing (VTOL) is desired. While not as fast and having a shorter range than their fixed-wing counterparts they can maintain a fixed-position hover. They are inherently stable as their center of mass (COM) is below their vertical thrust generating propeller. As the control of the helicopter uses a swash plate (shown in Fig. 2.2b), the helicopter can manoeuvre

without significantly changing the speed of its main rotor. This means that the helicopter is easier to scale up than a quadrotor, and is generally more efficient and it does not require the constant propeller velocity changes of a multirotor. The mechanical complexity of the helicopter however makes it expensive and prone to mechanical failure.

3. **Underactuated Multirotor UAVs** are currently among the most accessible to the general public, with becoming extensively used in cinematography, racing, small-scale infrastructure inspection and many other tasks. Their inherent mechanical simplicity compared to helicopters along with the benefit of VTOL operation make them a cheaper option. Furthermore they are much more agile due to their differential thrust control and inherent instability. They also have a smaller footprint than helicopters, and are more suitable for use in close contact environments. Like the two previous classes, they are an underactuated system with fewer control inputs than system outputs which can cause issues in some physical interaction tasks. A significant drawback compared to traditional helicopter UAVs is the energy efficiency, but also the scalability. Because the quadrotor is controlled through differential thrust, as the size of the UAV scales up, the rotor inertia becomes much more significant, making the control of larger models much more difficult.

4. **Omnidirectional Multirotors** were developed as an adaptation of underactuated multirotors to be better suited for aerial physical interaction. They are able to fully and independently control their position and orientation, allowing them to exert forces on the environment in any given pose. These UAVs are quite recent and specialized, with little public use, although commercial prospects such as contact-based inspection are rapidly developing, general for quasi-static tasks. The omni-directionality however comes at a cost. As the propellers do not share parallel thrust axes, they produce opposing thrust components which fight against one another. Along with the increased inter-propeller aerodynamic interference, this reduces the energy efficiency and the maximum speed. With highly effective camera gimbals now on the market, these UAVs are not necessary for most purely observational tasks, as the roll and pitch of a payload gimbal can compensate for the underactuation of the classical quadrotor.

As this thesis is primarily concerned with underactuated multirotors, a brief overview of

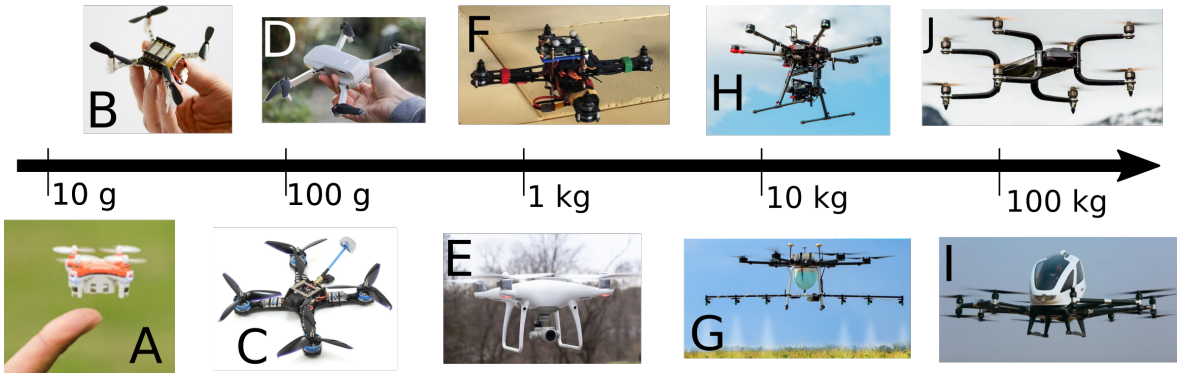


Figure 2.3 – A variety of quadrotor-like multirotor UAVs. Drone A is a cheap toy quadrotor. Drones B and F are quadrotors for scientific use. Drone C is a FPV racing quadrotor. Drones D and E are photography quadrotors. Drone G and H are hexarotors for crop spraying and professional cinematography respectively. Drone I is a 12-propeller multirotor for personal transportation and Drone J is a heavy-lift octorotor for transporting payloads up to 250 kg.

the current range of systems is provided: Current multirotors range from several grams to several hundred kilograms, as seen in Fig. 2.3. Toys for casual users are generally on the low end of the mass scale (10 g to 100 g) so that their propellers have very low inertia and are unlikely to harm a person. From 100 g and up to about 1 kg, multirotors (almost exclusively quadrotors) split into three main categories; first-person view (FPR) drones for racing and recreational flying, general use photography drones for non-cinematographic users such as real-estate professionals, and scientific drones used in labs. Above several kilograms, drones begin to get quite expensive and dangerous, and are generally limited to professional uses such as agricultural spraying, cinematography, payload delivery, and surveillance. These drones are often multirotors with 6, 8, 12, or 16 propellers to generate sufficient thrust without very large propellers. Multirotors that are heavier than several dozen kilograms are beginning to emerge as commercial prototypes for payload delivery, rescue, offshore, and human transportation application. These however are generally in the development stage and are not yet widely used.

2.2 Multirotor Modelling and Mechanics

In this section, we cover the modelling of multirotors, including the representation of their pose, their means of actuation, and their rigid-body mechanics used for later controllers.

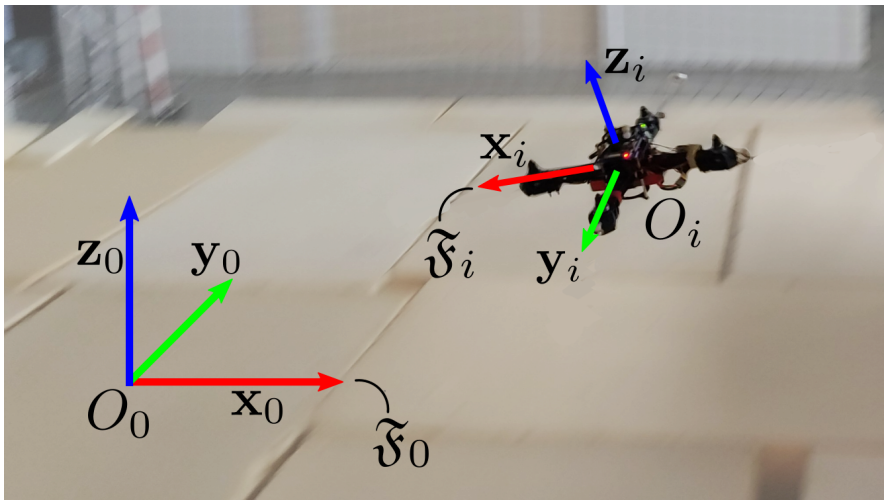


Figure 2.4 – A representation of the world frame $\mathfrak{F}_0$ and the drone frame $\mathfrak{F}_i$. Note that from any drone, $\mathfrak{F}_i$ may be arbitrarily rotated about $\mathbf{z}_i$, so long as it is by a fixed value. In this thesis, red, green, and blue are always used for the $\mathbf{x}$, $\mathbf{y}$, and $\mathbf{z}$ axes of a frame respectively.

2.2.1 Representation

In the most general sense, a multirotor is simply a UAV controlled by the differential actuation of fixed-pitch propellers. Considered as a rigid body, our generic multirotor i has a body-fixed local frame $\mathfrak{F}_i \left(O_i, \mathbf{x}_i, \mathbf{y}_i, \mathbf{z}_i\right)$ as shown in Fig. 2.4, which is taken as the reference frame for all local sensing information. Without the loss of generality, the origin O_i of $\mathfrak{F}_i$ is placed at the geometric center of the quadrotor structure, and at its center of mass (COM). With respect to some inertial frame $\mathfrak{F}_0 \left(O_0, \mathbf{x}_0, \mathbf{y}_0, \mathbf{z}_0\right)$, the position $\mathbf{p}_i = \vec{O_0 O_i}$ of the quadrotor is often measured in practice using longitude, latitude and altitude measurements $\in \mathbb{S}^2 \times \mathbb{R}^1$. As we are assuming formations to be located in a relatively small geographic area, a cartesian assumption $\mathbf{p}_i \in \mathbb{R}^3$ is used to define the position of the quadrotor [38].

Along with the position of the quadrotor, the rotation between $\mathfrak{F}_0$ and $\mathfrak{F}_i$ (often referred to as the "attitude") must be considered. The attitude representation of the quadrotor on $SO(3)$ is a matter of choice and the best-suited representation varies on a case-by-case basis, but in this manuscript there are three main representations used rotation matrices, Euler angles and quaternions. Not all aspects of each are presented here, but only the components of each which are necessary for following the work in this manuscript.

2.2.1.1 Rotation Matrices

The most intuitive representation, the columns of the rotation matrix contain the three orthogonal vectors of $\mathfrak{F}_i$

$$\mathbf{R}_i = \begin{bmatrix}\mathbf{x}_i & \mathbf{y}_i & \mathbf{z}_i\end{bmatrix} \tag{2.1}$$

expressed in $\mathfrak{F}_0$. This results in a mathematically simple manner of transforming vectors between reference frames. If the vector ${}^A\mathbf{x} \in \mathbb{R}^3$ is expressed in $\mathfrak{F}_A$, then the operation

$${}^B\mathbf{x} = {}^B\mathbf{R}_A\,{}^A\mathbf{x} \tag{2.2}$$

is sufficient for expressing the vector in the frame $\mathfrak{F}_B$, where ${}^B\mathbf{R}_A$ is the rotation matrix from $\mathfrak{F}_B$ to $\mathfrak{F}_A$. These rotations can be decomposed into elementary rotation matrices specifying the magnitude of a rotation θ about the $\mathbf{x}$, $\mathbf{y}$, or $\mathbf{z}$ axes of any given frame

$$\mathbf{R}_x(\theta) = \begin{bmatrix}1 & 0 & 0\\ 0 & \mathrm{c}\theta & -\mathrm{s}\theta\\ 0 & \mathrm{s}\theta & \mathrm{c}\theta\end{bmatrix} \quad \mathbf{R}_y(\theta) = \begin{bmatrix}\mathrm{c}\theta & 0 & \mathrm{s}\theta\\ 0 & 1 & 0\\ -\mathrm{s}\theta & 0 & \mathrm{c}\theta\end{bmatrix} \quad \mathbf{R}_z(\theta) = \begin{bmatrix}\mathrm{c}\theta & -\mathrm{s}\theta & 0\\ \mathrm{c}\theta & \mathrm{s}\theta & 0\\ 0 & 0 & 1\end{bmatrix} \tag{2.3}$$

Rotation matrices have certain interesting properties, the first of which is that each column and each row is a unit vector. Any valid rotation matrix satisfies $\text{rank}\,(\mathbf{R}) = 3$ as the columns by definition span an orthogonal basis of unit vectors. Operations inverting rotations are simple, as all rotation matrices satisfy ${}^B\mathbf{R}_A = {}^A\mathbf{R}_B^{-1} = {}^A\mathbf{R}_B^T$, and thus matrix inversions may be avoided by a simple transpose.

2.2.1.2 Euler Angles

This is a parameterization of the rotation of $\mathfrak{F}_i$ by three successive rotations about its own axes. From the initial orientation $\mathbf{R}_i = \boldsymbol{\mathcal{I}}_3$, the frame $\mathfrak{F}_i$ is first rotated by an angle ψ_i about one axis, then by an angle θ_i about another axis, and finally by an angle ϕ_i about another axis. In this dissertation, whenever Euler angles are used they respect the Z-Y-X (yaw-pitch-roll) convention. Therefore the rotation matrix of $\mathfrak{F}_i$ can be expressed as

$$\mathbf{R}_i = \mathbf{R}_z(\psi_i)\mathbf{R}_y(\theta_i)\mathbf{R}_x(\phi_i) \tag{2.4}$$

While Euler angles give an intuitive representation of attitudes, they are very

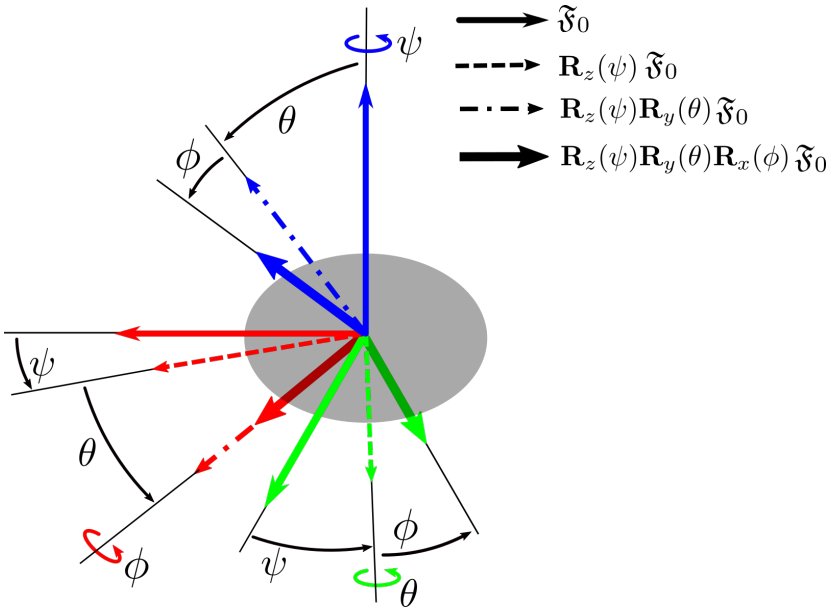


Figure 2.5 – A representation of the orientation of frames by a ZYX Euler angles

computationally inefficient. Furthermore, they are subject to singularities (known as gimbal lock) that may cause issues in very aggressive flights. The primary advantage is that they permit a decoupling between the components of the attitude normal to and parallel to the thrust axis of the quadrotor which will be important in section 2.3.3 when discussing flat outputs.

2.2.1.3 Quaternions

Quaternions are often found to be hard to visualize, but are used extensively for orientation representation in robotics, aviation, computer graphics and more, as they do not have singularities and are very computationally efficient. The quaternion $\mathbf{q}(q_w, q_x, q_y, q_z)$ describes the structure

$$\mathbf{q} = q_w + q_x\hat{\mathbf{i}} + q_y\hat{\mathbf{j}} + q_z\hat{\mathbf{k}} \tag{2.5}$$

where q_w, q_x, q_y, and q_z are real numbers, and $\hat{\mathbf{i}}$, $\hat{\mathbf{j}}$, and $\hat{\mathbf{k}}$ are complex numbers defined by $\hat{\mathbf{i}}^2 + \hat{\mathbf{j}}^2 + \hat{\mathbf{k}}^2 = \hat{\mathbf{i}}\hat{\mathbf{j}}\hat{\mathbf{k}} = -1$. This representation can be thought of as a real rotation (related to q_w) about an axis (defined by q_x, q_y, and q_z). Despite the rather intimidating structure of quaternions, they can greatly simplify orientation math. In our case, we deal exclusively with unit quaternions having a norm given by

$$||\mathbf{q}|| = \sqrt{q_w^2 + q_x^2 + q_y^2 + q_z^2} = 1 \tag{2.6}$$

Consider that we have two rotations $\mathbf{R}_1$ and $\mathbf{R}_2$, represented by the quaternions $\mathbf{q}_1$ and $\mathbf{q}_2$ respectively. The combined rotation $\mathbf{R}_1\mathbf{R}_2$ is given as the (non-commutative) product of the two quaternions $\mathbf{q}_1 \otimes \mathbf{q}_2 = \mathbf{Q}(\mathbf{q}_1)\mathbf{q}_2$

$$\mathbf{q}_1 \otimes \mathbf{q}_2 = \underbrace{\begin{bmatrix} q_{1,w} & -q_{1,x} & -q_{1,y} & -q_{1,z} \\ q_{1,x} & q_{1,w} & -q_{1,z} & q_{1,y} \\ q_{1,y} & q_{1,z} & q_{1,w} & -q_{1,x} \\ q_{1,z} & -q_{1,y} & q_{1,x} & q_{1,w} \end{bmatrix}}_{\mathbf{Q}(\mathbf{q}_1)} \mathbf{q}_2 \tag{2.7}$$

The conjugate of a quaternion can be seen as a real rotation around the negative imaginary axis, resulting in the reverse rotation. Therefore the quaternion conjugate is simply

$$\mathbf{q}^* = [q_w \ -q_x \ -q_y \ -q_z]^T \tag{2.8}$$

As we deal exclusively with unit quaternions, the inverse of a quaternion is simply equal to its conjugate, although generally the quaternion inverse can be found as

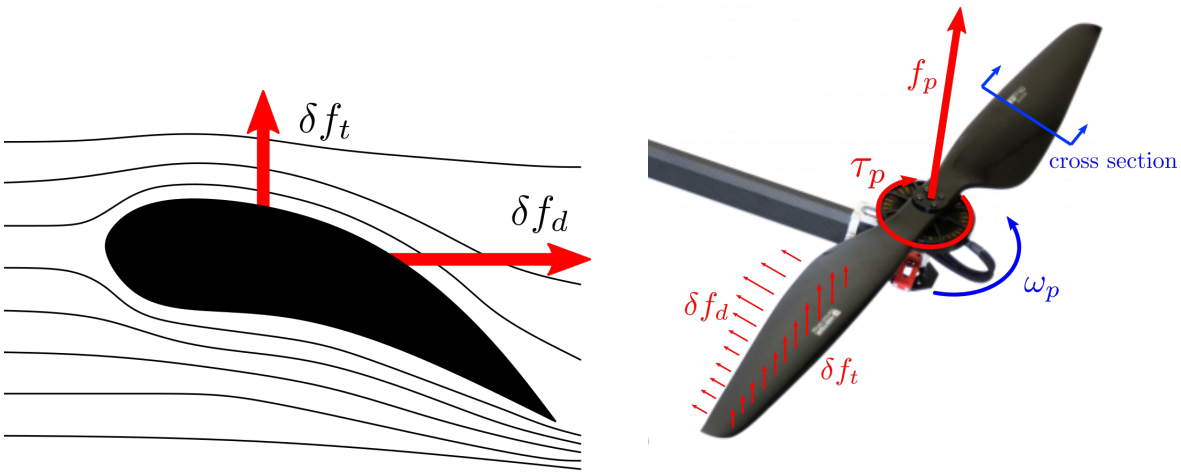
$$\mathbf{q}^{-1} = \frac{\mathbf{q}^*}{||\mathbf{q}||^2} \tag{2.9}$$

The derivative of a quaternion can be found as a function of the angular velocity measured in its own frame, or in the reference frame. For quadrotors, which are always equipped with a gyroscope, the angular velocity $\boldsymbol{\omega}$ in the body frame is directly measured, thus the quaternion derivative can be calculated as

$$\dot{\mathbf{q}} = \frac{1}{2} \begin{bmatrix} 0 \\ \boldsymbol{\omega} \end{bmatrix} \otimes \mathbf{q} \tag{2.10}$$

2.2.2 Actuation Model

Now that we have shown how to represent the frame of a multirotor, the dynamic model is developed which expresses the relationship between the forces acting on the multirotor and its motion in $SE(3)$. The actuation of multirotors comes from forces and torques exerted by the propellers as shown in Fig. 2.6 which are generated by setting the angular velocity Ω_p of the propeller [39]. Each propeller is a rotating aerofoil that produces lift forces (f_p) and drag torques (τ_p) as it moves through the air, and while the aerodynamic details will not be treated



(a) A cross section of the propeller showing the airflow around the propeller and the differential thrust and drag forces acting on the propeller section

(b) The propeller of a multirotor annotated with the aerodynamic thrust force f_p and drag torque τ_p acting on the propeller

Figure 2.6 – Propeller actuation diagrams

in this dissertation, they can be found in [40]. In aerial robotics, the relatively complicated aerodynamics are often simplified to

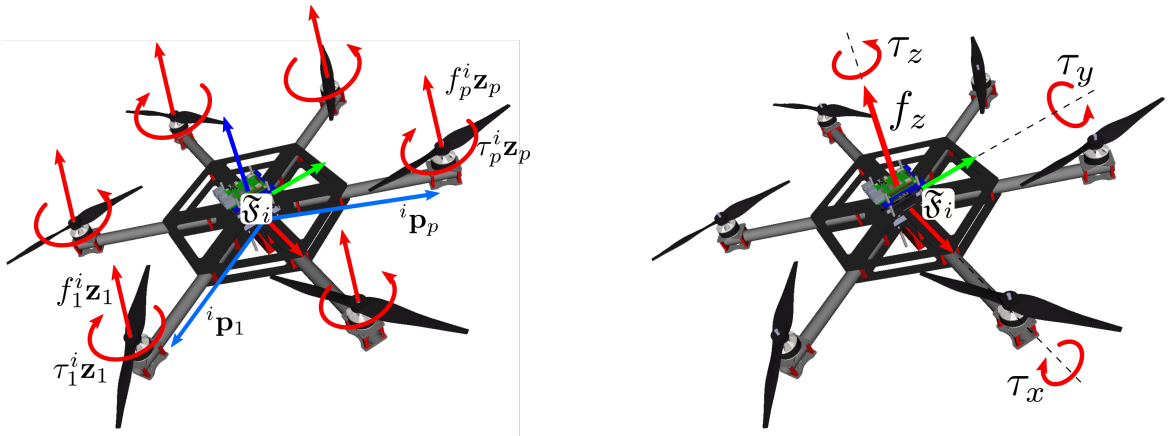
$$f_p = k_t \Omega_p^2 \tag{2.11}$$

$$\tau_p = -\text{sign}(\Omega_p) k_d \Omega_p^2 \tag{2.12}$$

where k_t and k_d are the propeller coefficients of thrust and drag. This neglects forces such as the ground effect (which occurs when the multirotor is less than one rotor length from the ground [41]) and rotor flapping (at high apparent wind speeds [42]) but are reasonably accurate otherwise. These coefficients may be identified either with a dedicated test bench, or with an identification based on in-flight data [13], [43], [44]. The control of the propeller speed is set by the electronic speed controllers and may be approximated as a first order system [45], although in practice for small UAVs this is often disregarded.

Each propeller is mounted to the multirotor frame at position ${}^i\mathbf{p}_p$ with rotational axis ${}^i\mathbf{z}_p$ expressed in $\mathfrak{F}_i$ (see Fig. 2.7). Rotating at a given angular velocity to generate f_p and τ_p, the propeller generates the wrench ${}^i\mathbf{w}_p$ as shown in Eq. (2.13), expressed in $\mathfrak{F}_i$.

$${}^i\mathbf{w}_p = \begin{bmatrix} \mathbf{f}_p \\ \boldsymbol{\tau}_p \end{bmatrix} = \begin{bmatrix} f_p {}^i\mathbf{z}_p \\ f_p {}^i\mathbf{z}_p \times {}^i\mathbf{p}_p + \tau_p {}^i\mathbf{z}_p \end{bmatrix} \tag{2.13}$$



(a) All actuation forces and torques acting on the hexarotor.

(b) The linearly independant hexarotor inputs

Figure 2.7 – The actuation of an underactuated hexarotor.

For quadrotor-like UAVs, the axes of all the propellers are parallel to the z-axis of $\mathfrak{F}_i$. This is the case in many commercial multi-rotors even if they have 6, 8, or more propellers, because when all the propellers share a common axis, there is no energy lost by propellers fighting against each other [46]. The downside of this is that the quadrotor cannot exert a force orthogonal to the local z-axis and thus for aerial interaction tasks, omnidirectional multirotors are gaining popularity [35], [36], [47]. This thesis however limits itself to quadrotor-style multirotors, therefore the wrench a single propeller exerts on $\mathfrak{F}_i$ is

$$ {}^i\mathbf{w}_p = \begin{bmatrix} f_x \\ f_y \\ f_z \\ \tau_x \\ \tau_y \\ \tau_z \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ f_p \\ f_p \, {}^i\mathbf{p}_p^T \mathbf{e}_2 \\ -f_p \, {}^i\mathbf{p}_p^T \mathbf{e}_1 \\ \tau_p \end{bmatrix} \tag{2.14} $$

where the vector $\mathbf{e}_x$ is a vector of zeros except for the x^{th} element which is 1 (e.g. $\mathbf{e}_2 = [0\ 1\ 0]^T$). The actuation wrench of the multirotor is the sum of the wrenches of the individual

propellers expressed in $\mathfrak{F}_i$. This is often expressed in the form

$$\underbrace{\begin{bmatrix} f_z \\ \tau_x \\ \tau_y \\ \tau_z \end{bmatrix}}_{\mathbf{w}_a} = \underbrace{\begin{bmatrix} k_t & \cdots & k_t \\ k_t\,{}^i\mathbf{p}_1^T\mathbf{e}_2 & \cdots & k_t\,{}^i\mathbf{p}_p^T\mathbf{e}_2 \\ -k_t\,{}^i\mathbf{p}_1^T\mathbf{e}_1 & \cdots & -k_t\,{}^i\mathbf{p}_p^T\mathbf{e}_1 \\ -k_d\text{sign}(\Omega_1) & \cdots & -k_d\text{sign}(\Omega_p) \end{bmatrix}}_{\mathbf{\Gamma}} \underbrace{\begin{bmatrix} \Omega_1^2 \\ \vdots \\ \Omega_p^2 \end{bmatrix}}_{\mathbf{\Omega}} \tag{2.15}$$

where $\mathbf{w}_a$ is the actuation wrench of the multirotor in $\mathfrak{F}_i$, $\mathbf{\Omega}$ is the column vector of squared propeller speeds and $\mathbf{\Gamma}$ is the constant mixer matrix relating the two values. It can be seen that regardless of the number of propellers (assuming a minimum of four), the actuation wrench can be expressed as a single thrust force in the $\mathbf{z}_i$ direction of the multirotor and three orthogonal torques acting on the multirotor. Furthermore, given a desired actuation wrench, one can find the corresponding propeller velocity inputs by solving for propeller speeds. Feasible actuation wrenches must of course respect the fact that all propeller velocities are either strictly positive or strictly negative, and has a finite-bounded real value. For a quadrotor, a given feasible actuation wrench has a single solution obtained by $\mathbf{\Omega} = \mathbf{\Gamma}^{-1}\mathbf{w}_a$. Multirotors in general (such as underactuated hexarotors, octorotors, etc...) can have an infinite set of solutions defines by $\mathbf{\Omega} = \mathbf{\Gamma}^{\dagger}\mathbf{w}_a + \ker(\mathbf{\Gamma}^T)\boldsymbol{\lambda}$ where $\boldsymbol{\lambda} \in \mathbb{R}^{\text{rank}(\ker(\mathbf{\Gamma}))}$ is used to distribute the $p-4$ redundant propeller velocities [48]–[50] such that Eq. (2.15) is satisfied, but nonetheless only produces the 4-dimensional wrench $\mathbf{w}_a$. This underactuation (the control input $\mathbf{u}_i \in \mathbb{R}^4$ is insufficient to fully actuate a rigid body in $SE(3)$) is one of the principle drawbacks of quadrotor-style multirotors, as it introduces additional challenges in the control. Nonetheless it is also beneficial because of the increased power efficiency compared for omnidirectional UAVs.

2.2.3 Rigid-Body Dynamic Model

The rigid-body dynamic model of multirotors can be developed from the Newton-Euler formulation relating the force $\mathbf{f}_i$ and torque $\boldsymbol{\tau}_i$ acting on a body with mass m_i and inertia tensor $\mathbf{I}_i$ to its linear acceleration $\ddot{\mathbf{p}}$ and angular acceleration $\dot{\boldsymbol{\omega}}_i$. This is simplified to

$$\begin{bmatrix} \mathbf{f}_i \\ \boldsymbol{\tau}_i \end{bmatrix} = \begin{bmatrix} m_i\boldsymbol{\mathcal{I}}_3 & \mathbf{0} \\ \mathbf{0} & \mathbf{I}_i \end{bmatrix} \begin{bmatrix} \ddot{\mathbf{p}}_i \\ \dot{\boldsymbol{\omega}}_i \end{bmatrix} + \begin{bmatrix} \mathbf{0} \\ \boldsymbol{\omega}_i \times \mathbf{I}_i\boldsymbol{\omega}_i \end{bmatrix} \tag{2.16}$$

by the prior assumption that $\mathfrak{F}_i$ is located at the multirotor COM, and that the mass of the quadrotor is symmetrically distributed. For details in the contrary case, the reader may refer to [13], but the differences within the scope of this thesis are negligible and may be compensated for if eventually significant (for example if the UAVs are transporting large off-center payloads).

Considering only the translational component of Eq. (2.16) in $\mathfrak{F}_0$, the multirotor i may be considered as a point mass experiencing an actuation force of $f_i\mathbf{z}_i$, a force due to gravity of $\mathbf{g} = [0\ 0\ -9.81]$ ms^{-2} and an acceleration $\dot{\mathbf{v}}$, all expressed in $\mathfrak{F}_0$. The quadrotor has a velocity $\mathbf{v}_i$ which is the derivative of it's position in $\mathfrak{F}_i$. The translational state of the multirotor is therefore modelled as

$$\dot{\mathbf{p}}_i = \mathbf{v}_i \tag{2.17a}$$

$$\dot{\mathbf{v}}_i = \frac{1}{m_i} f_i \mathbf{z}_i + \mathbf{g} \tag{2.17b}$$

This of course neglects the fact that the quadrotor can both reorient the direction of $\mathbf{z}_i$ and rotate around $\mathbf{z}_i$. The angular acceleration of $\mathfrak{F}_i$ expressed in $\mathfrak{F}_i$ can be calculated from the Newton-Euler formulation in Eq. (2.16). The attitude rate of change in $\mathfrak{F}_0$ can then be calculated from both the current attitude and the angular velocity expressed in $\mathfrak{F}_i$. This gives the attitude model

$$\dot{\mathbf{R}}_i = \mathbf{R}_i[\boldsymbol{\omega}_i]_\times \tag{2.18a}$$

$$\dot{\boldsymbol{\omega}}_i = \mathbf{I}_i^{-1}(\boldsymbol{\tau}_i - [\boldsymbol{\omega}_i]_\times \mathbf{I}\boldsymbol{\omega}_i) \tag{2.18b}$$

were $[]_\times$ is the skew symmetric matrix operator.

The model presented is that of a perfectly rigid and symmetrical quadrotor operating in perfectly still air, thus it will not be perfectly accurate due to numerous factors. Static offsets such as a non-centred COM or slightly misaligned propeller axes, and dynamic effects such as aerodynamic drag and propeller flapping will perturb the model so a robust closed-loop control is required for steady flight, as discussed in the following section. Some of these unidentified factors such as aerodynamic drag may be identified and taken into account by Eqns 2.17-2.18 with either physics-based [51] or learning-based approaches [52], but the model accuracy of quadrotors is insufficient to have open loop stability for any significant length of time.

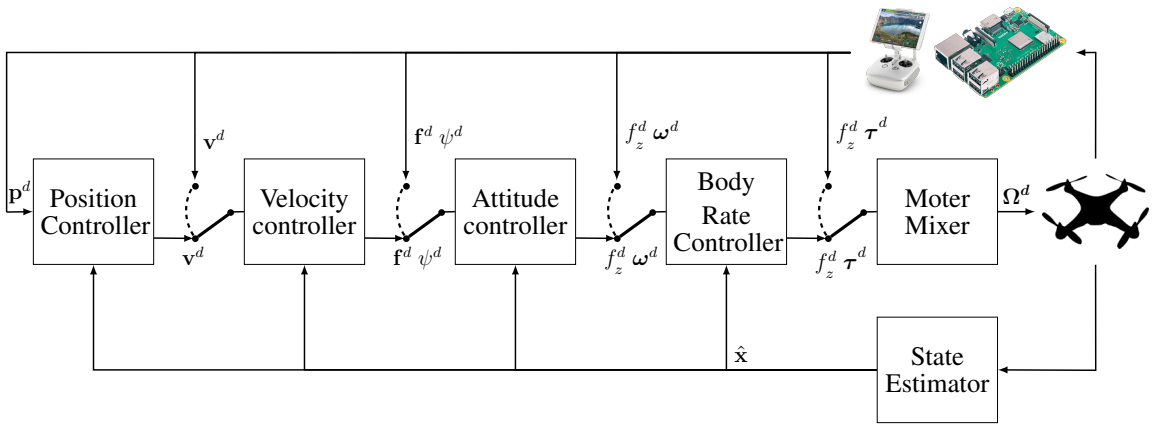


Figure 2.8 – A general multirotor control diagram, showing the different levels of control. All the controllers receive estimated state feedback from the state estimator, and calculates a reference trajectory for the lower-level controllers.

2.3 Multirotor Control

The control of multirotors has been extensively studied and many methods proposed. They all however must account for the underactuation of the quadrotor, and generally do so through a cascade architecture (see Fig. 2.8) where an outer controller is used to calculate references for an inner controller, which then outputs the required actuation. In a more practical sense, the translational controller(s) calculates the direction the quadrotor must move, and generates a reference attitude and thrust that would allow the multirotor to move in that direction. The attitude controller(s) then produces an actuation wrench using the propellers to track the reference attitude. This dissertation does not detail all quadrotor controllers, which (non-exhaustively) include feedback linearization, sliding mode, Lyapunov and model predictive control, however it is important to go over at least the basics of the requisite multirotor-level control components.

2.3.1 Translational Controllers

The goal of the translation controller (frequently referred to as the "outer loop") is to make the multirotor follow a desired cartesian velocity set by a high-level planner. This could be in the form of positional waypoints, a reference velocity output from a formation controller, or even joystick input.

A position controller attempts to steer the quadrotor to a position $\mathbf{p}^f$. To produce a smoother flight however, a trajectory over a timespan $t \in [0, t_f]$ is often used. Without getting into details (see [53] for more) a continuous position $\mathbf{p}^d(t)$, velocity $\mathbf{v}^d(t)$, and

acceleration $\dot{\mathbf{v}}^d(t)$ trajectory may be generated. The goal of the controller is then to act on the quadrotor such that in a finite time the translational state of the drone $\mathbf{q} = [\mathbf{p}, \mathbf{v}, \dot{\mathbf{v}}]$ approaches the desired translational state $\mathbf{q}^d = [\mathbf{p}^d, \mathbf{v}^d, \dot{\mathbf{v}}^d]$. The control vector $\mathbf{f} = f_z \mathbf{z}$ may be computed using a feedback linearization controller acting on Eq.(2.18b) with either a proportional derivative (PD) or proportional-integral-derivative (PID) control law with acceleration feedforward (FF). This may take the form of

$$\mathbf{f} = m \left(\underbrace{k_p \left(\mathbf{p}^d - \mathbf{p} \right)}_{\text{P}} + \underbrace{k_d \left(\mathbf{v}^d - \mathbf{v} \right)}_{\text{D}} + \underbrace{k_i \int_0^t \left(\mathbf{p}^d - \mathbf{p} \right) \mathrm{d}t}_{\text{I}} + \underbrace{\dot{\mathbf{v}}^d}_{\text{FF}} - \mathbf{g} \right) \tag{2.19}$$

where k_p, k_d and k_i are strictly positive gains (apart from PD controllers where $k_i = 0$).

Another popular method which reduces the need for smooth trajectories (and is used in the popular PX4 flight stack) is to use a simple proportional control law to control the position

$$\mathbf{v}^d = k_p \left(\mathbf{p}^d - \mathbf{p} \right) \tag{2.20}$$

which is then saturated to respect velocity limits. This value is passed as a reference to a velocity controller which uses a PID controller to calculate the desired control thrust vector minimizing the velocity error $\mathbf{e}_v = \mathbf{v}^d - \mathbf{v}$ as

$$\mathbf{f} = m\mathbf{g} + k_p \mathbf{e}_v + k_d \dot{\mathbf{e}}_v + \int_0^t \mathbf{e}_v \mathrm{d}t \tag{2.21}$$

The advantage of decoupling the position and velocity controls as such is that different control modes may make use of the same lower-level gain tuning and control code.

2.3.2 Attitude Controllers

While the translational controllers work on the $\mathbb{R}^3$ manifold, the attitude controllers manoeuvre the quadrotor on the $SO(3)$ manifold to both orient $\mathbf{z}_i$ in the desired direction (roll and pitch), and to maintain the desired yaw (constraining the rotation around the $\mathbf{z}_i$ axis). The exact formulation of the attitude control law depends on the attitude representations used, however in general there is a control component proportional to the attitude error driving the system to the desired spatial orientation, and a control law acting on the angular rate providing damping to the system.

We may assume that some command (coming from a user or some higher-level controller) sends the multirotor a reference thrust force $\mathbf{f}^d$ and yaw ψ^d. The challenge then becomes to reconstruct some desired attitude $\mathbf{q}^d$ for the UAV such that $\mathbf{z}_i^d$ is parallel to $\mathbf{f}^d$ and then rotate the UAV around it's $\mathbf{z}_i^d$ axis as defined by the desired yaw. The first step can be written as

$$\mathbf{z}_i^d = \frac{\mathbf{f}^d}{||\mathbf{f}^d||} \tag{2.22}$$

which leads to the rotation matrix

$${}^0\mathbf{R}_i = [\mathbf{x}_i(\psi^d)\ \mathbf{y}_i(\psi^d)\ \mathbf{z}_i^d] \tag{2.23}$$

There are then multiple methods which may fix the yaw, either by geometrically selecting unit vectors as done in [54], or by using Euler angles as in [55]. In either case, the result is a fully defined desired attitude, controllable in any form of representation. We present here the quaternion method, as it is computationally efficient, and stable anywhere on $SO(3)$.

The error between the desired and mesured quaternion can be expressed as

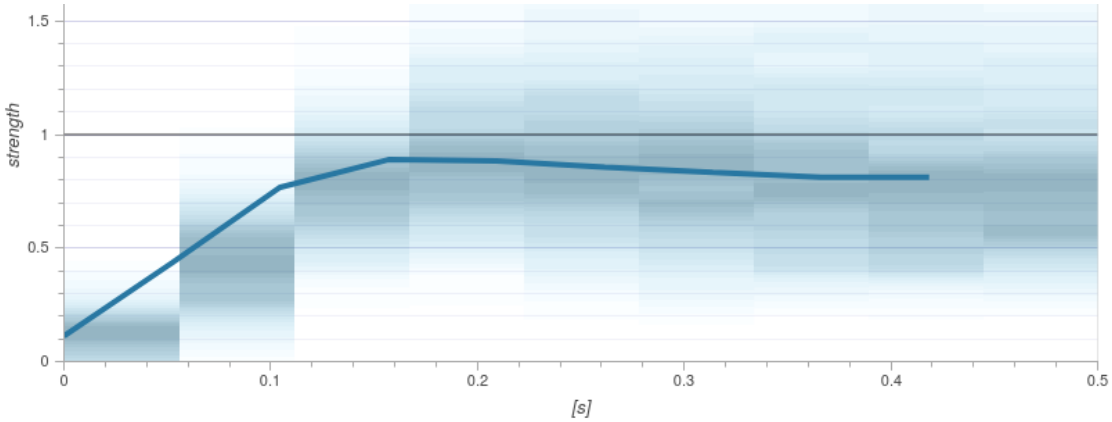
$$\mathbf{e}_\mathbf{q} = \mathbf{q}^d \otimes \mathbf{q}^{-1} \tag{2.24}$$

which, recalling the definition of a quaternion, defines the rotation from the current attitude to the desired attitude a a rotation $\mathbf{e}_{\mathbf{q},w}$ around the axis defined by $\mathbf{e}_{\mathbf{q},xyz}$. A controller can therefore be defined to create a moment-based PD controller on the attitude and angular velocity

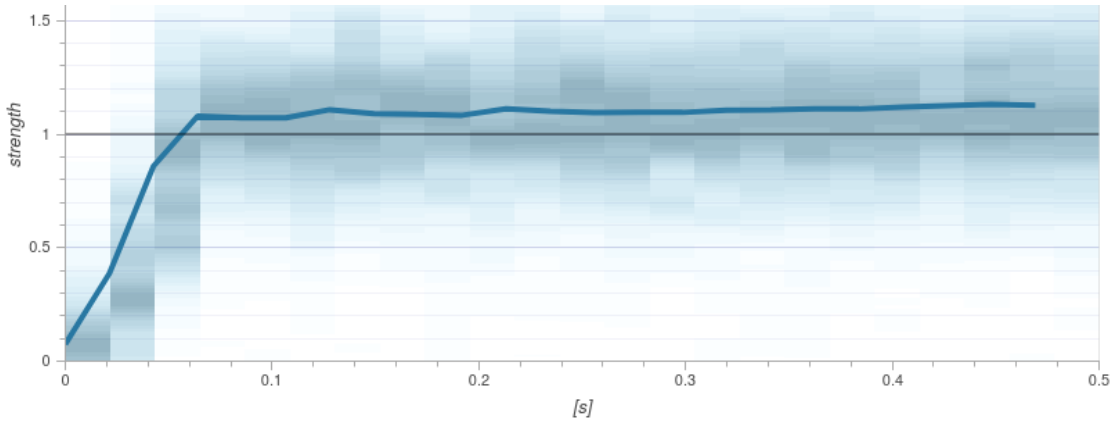
$$\boldsymbol{\tau} = k_p \text{sign}(\mathbf{e}_{\mathbf{q},w})\mathbf{e}_{\mathbf{q},xyz} - k_d\boldsymbol{\omega} \tag{2.25}$$

Similar to the translational control, an alternative method may decouple the attitude and attitude rate components of the attitude control. The attitude error component of the attitude control can be used the generate a desired angular velocity $\boldsymbol{\omega}^d$ which may then has a separate body-frame angular velocity control loop (often using PD or PID control) to track the reference angular velocity.

It is important to be aware of the dynamic response of quadrotor control to understand the reasoning and assumptions made later on. We have shown how generally quadrotor controllers are decomposed into two to four loops which may receive their reference command from either the preceding controller or an external source. To achieve good



(a) The roll response



(b) The roll rate response

Figure 2.9 – Normalized response for the roll and roll-rate dynamics of a quadrotor. The blue line shows the best fit, and the shaded color bars show the response distribution of many different manoeuvres at a given time interval. These graphs were generated from PX4 Flight Review tool using flight data from one of our in-house experimental quadrotors.

performance, each of the controllers must be able to accurately track the reference command. This implies that each controller must have faster dynamics than the controller that precedes it, otherwise it may not be able to track the reference trajectory. This is best seen in the attitude and attitude rate responses for a quadrotor which was aggressively flown in the flight arena. Figure 2.9 show an analysis of the roll and roll rate response of the quadrotor during an aggressive flight, and it can be seen the attitude rate controller converges after around 50 ms, and the attitude control after about 150 ms. A similar relationship is true for the translational components of the controller, with the velocity response being on the order of a second, and the positional controller responding even slower. This is in fact one of the reasons that many skilled drone-racing pilots use rate control modes (i.e. directly map the joysticks to thrust and angular velocity).

2.3.3 Differential Flatness for Quadrotors

While hitherto we have discussed how the use of backstepping controllers may be used to control a quadrotor despite its underactuation, we have not yet discuss what trajectories it may follow. Clearly with four inputs it is unable to track an arbitrary trajectory in $SE(3)$, so what are the limitations on the trajectories that it is in fact able to follow? Differential flatness is a property of a system which permits the state and control inputs to be determined as a function of a set of variables (called flat outputs) and their derivatives [56]. Anyone who has piloted a quadrotor is able to tell that the pilot has four inputs (or joystick motions), corresponding to the three translational acceleration and the UAV yaw rate, with the roll and pitch being coupled to the horizontal translational acceleration. This lead to the reasonable hypothesis that a good choice of flat output for a quadrotor is

$$\mathbf{x} = [p_x \; p_y \; p_z \; \psi]^T \in \mathbb{R}^3 \times \mathbb{S}^1 \tag{2.26}$$

It has long since been mathematically proven that these are indeed flat outputs of a quadrotor [54]. Indeed, it is furthermore shown that when considered rotor drag, the primary aerodynamic effect not included in the standard quadrotor model we have presented, that the same set of flat outputs hold true [51]. The inputs of the quadrotor can therefore be shown to be functions of the flat outputs and their derivatives, with the thrust and yaw moments being functions of the second derivative of $\mathbf{x}$, while the roll and pitch moments are functions of the

fourth derivative. This means that a quadrotor may exactly track a trajectory $\mathbf{x}(t)$ so long as $\mathbf{x}^{(4)}(t)$ is continuous.

2.4 Multirotor Sensing and Estimation

So far only the states of the multirotors are discussed, without indication of their measurement and estimation. In this section, basic onboard sensors to reconstruct the UAV's states are discussed, and then details on various exteroceptive sensors used to measure interaction with the environnement are presented.

2.4.1 Proprioceptive Sensing and State Estimation

Every multirotor must have a state estimation algorithm, as all states are neither measured directly, nor concurrently. While quadrotors can carry a variety of sensors and have different data fusion algorithms, every multirotor has at the very least an inertial measurement unit (IMU) containing at least a gyroscope and and accelerometer [42]. The basic requirements of a multirotor state estimator are to take the accelerometer and gyroscope readings, and output an estimated state consisting of at minimum $\hat{\mathbf{x}} = [\mathbf{v}, \mathbf{q}, \boldsymbol{\omega}]$. We remark that $\dot{\mathbf{v}}$ and $\boldsymbol{\omega}$ are directly measured but often subjected to low-pass filtering. The velocity of the multirotor may also be measured using sensors such as optical flow, and the position in the world frame is frequently added as a state, making use of some external (often GPS, SLAM, or motion capture) measurements. It is very important to remark that the yaw angle of the attitude is unconstrained. Magnetometers (3-axis electronic compasses) may be used to provide a measurement, however this are often impresses and easily biased by onboard electronics or environmental magnetic fields.

The state estimation used on the drones in our lab uses of a 24-state extended Kalman filter (EKF), which along with the minimum set of state estimates, also estimates magnetic field information, horizontal wind velocity, position, and sensor bias. While the details are outside the scope of this thesis, the general principal is that at some iteration $k+1$ of the estimator, the dynamic model of the quadrotor is used to create a update the prediction based on the previous prediction k and the sensor measurements taken at k

$$\hat{\mathbf{x}}_{k+1} = \dot{\mathbf{x}}_k(\mathbf{x}_k, \mathbf{u}_k)\Delta t \tag{2.27}$$

The uncertainty due to measurement bias and noise is propagated to the covariance matrix, which will grow over the iterations as the algorithm integrates error. When measurement $\mathbf{z}_{k+1}$ giving a direct mapping to a state (a GPS signal for example) is received, the estimate is updated as

$$\hat{\mathbf{x}}_{k+1} = \hat{\mathbf{x}}_{k+1} + \mathbf{k}_e \left(\mathbf{x}(\mathbf{z}_{k+1}) - \hat{\mathbf{x}}_{k+1} \right) \tag{2.28}$$

where $\mathbf{k}_e$ is an estimator gain. The uncertainty of the measurement is propagated to the state covariance matrix, generally reducing the uncertainty accumulated by integration in the prediction phase.

2.4.2 Exteroceptive Sensors

Exteroceptive sensors are generally used to measure information related to the robot's environnement, including humans and other robots. Some exteroceptive sensors such GPS allow the robot to measure its position in some absolute reference frame, while others such as cameras are often used to recover information relative to the robot's reference frame. Here we briefly discuss distance, bearing, and position sensors, with a focus on bearing measurements.

2.4.2.1 Distance Sensors

Distance sensors have been used in multi-robot control, however in practice, distance-only measurements are difficult to recover between individual robots. Most often using ultrasound or lasers, the sensor emits a signal an measures the time taking for the sound wave of laser to reflect off a surface and return to the sensor. The former typically measures the distance of an object along a wide arc, while the later measures the distance along a narrow beam. Range finders are generally unable to distinguish between detected objects, thus are most useful for detecting large objects such as walls, the ground (often used in altitude estimation), and large obstacles to be avoided. As such, while many works have made use of them for multi-robot formation control, they are not a practical solution for formation control in real situations (apart from obstacle avoidance). Of course, there are some sensors which measure relative position from which distances may be calculated as

$$d_{ij} = ||\mathbf{p}_j - \mathbf{p}_i|| \in \mathbb{R}^1 \tag{2.29}$$

where $\mathbf{p}_j$ is the position of the measured point and $\mathbf{p}_i$ is the position of sensor, both in $\mathfrak{F}_i$. In the case where relative positions are available however, position-based formation controllers would make a more appropriate choice than distance-based controllers.

2.4.2.2 Bearing Sensors

Bearing measurements make up the core of this thesis, and are simply a measurement of relative direction and have been used long before the rise of robotic. In maritime navigation before the electronics were invented, bearing triangulation with respect to landmarks was used to locate vessels at sea, and the bearing with respect to magnetic north (or even celestial features) was used to control the direction of travel. Bearing formation controllers were initially less studied than distance-based formation controllers, however became more popular when light, cheap, and information-rich bearing sensors became available. This sensor is of course the monocular camera, and while other bearing sensors such as antenna arrays exist, the camera is the one considered in this dissertation.

While "bearing" simply means a direction, in this manuscript we consider in to be measured in a robot's local frame $\mathfrak{F}_i$. This means that the bearing of some object at position $\mathbf{p}_j$ in $\mathfrak{F}_0$ measured by a multirotor i in $\mathfrak{F}_i$ can be calculated as

$$\boldsymbol{\beta}_{ij} = \mathbf{R}_i^T \frac{\mathbf{p}_j - \mathbf{p}_i}{d_{ij}} \in \mathbb{S}^2 \tag{2.30}$$

and correspond to relative position between the sensor and the measured object, projected onto the unit sphere around $\mathfrak{F}_i$. This may be done through an identified camera so long as the point of interest (in this thesis, the center of another multirotor $\mathfrak{F}_j$) is detected in the image. More details about this may be found in appendix B.

2.4.2.3 Position Sensors

The two primary technologies allowing for relative position sensing are vision systems (including the popular RGB-D cameras) and laser sensors (lidars). Stereo cameras take two pictures of the environment from two different cameras of known offset, and use this information to calculate the depth of points in the scene. Lidar sensors are effectively an array of distance sensors which rotate while collecting distance data, forming a 3D point cloud of surfaces around the sensor. The relative displacement between two points in $\mathfrak{F}_0$ is

(a) A quadrotor carrying a camera

(b) A quadrotor carrying a lidar

Figure 2.10 – A qualitative comparison of the field of view (highlighted in red) of lidars and cameras

expressed as $\mathbf{p}_{ij} = \mathbf{p}_j - \mathbf{p}_i$, however in practice it may be measured in $\mathfrak{F}_i$ as

$$\mathbf{p}_{ij} = \mathbf{R}_i^T(\mathbf{p}_j - \mathbf{p}_i) \in \mathbb{R}^3 \tag{2.31}$$

While relative position measurements are more information-rich (measuring 3 DOF compared to 2 DOF for bearings and 1 DOF for distance) they require more expensive and heavier (a major factor in aerial robotics) sensors which also require more computational power (thus heavier, more expensive computers) to process. A high quality lightweight lidar such as the Velodyne Puck Lite weighs 600 g, has a $30°$ high field of view, and costs on the order of 5000 €. On the other hand, a fisheye camera with a $160°$ high field of view may weight as little as 20 grams and be found for around 50 €. It is therefore of great interest to be able to be able to maintain formations with just bearing measurements instead of requiring full relative positioning.

CHAPTER 3

INTRODUCTION TO MULTI-ROBOT SYSTEMS

MULTI-ROBOT systems have the potential to augment the capabilities of individual robots, without the additional cost and practical limits of scaling up the speed or strength of the robots. In this chapter, we first present the interest in using multi-agent systems for a variety of applications, and then focus on the existing state of the art for aerial multi-robot systems. Key concepts such as decentralization are discussed, along with the mathematical tools such as graph theory that are required. The main types of multi-robot control are presented, and formation control is discussed in detail. The different agent models and inter-agent measurements are presented with their respective mathematical and

technological challenges, and a thorough theoretical review of bearing formation control is provided.

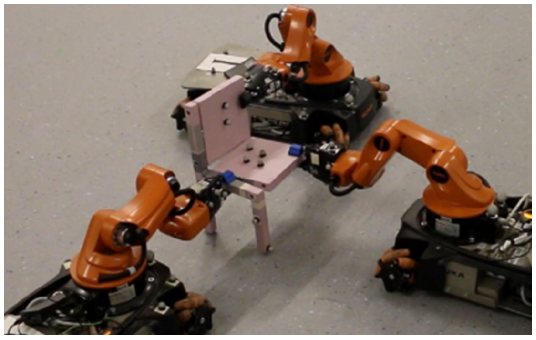
3.1 An Introduction of Multi-Robot System

A multi-robot system is a group of robots each with individual sensing, actuation, and computing capabilities that interact with one another with some degree of autonomy. Often described as multi-agent systems, the difference in nomenclature is generally semantic and depends on the community (i.e. computer scientists will likely use “Agents" systems, while controls and hardware oriented researchers will talk about “Robots”). In this thesis they are interchangeable, but we attempt to keep consistency between chapters and a more detailed discussion of the differences may be found in [57].

3.1.1 Motivation

Multi-robot systems have proven applications, primarily in the automotive industry, where the versatility of many smaller robots cannot be achieved with a single better-performing module [58]. During the initial assembly-line use of robots, each robot would perform a single task in a serial manner, just as a human worker would. Particularly for large tasks such as automotive assembly however, it is faster for robots to work in parallel, in which case each robot needs to account for the presence of other robots, as well as task constraints (e.g. robot B needs robot A to complete a task before it can begin its own task) [59]. This has further evolved into collaborative tasks where some robots help others to better accomplish their tasks by carrying and orienting parts [60]. Now however, we are interested in robots leaving the highly ordered environments of manufacturing cells, and want robots to collaborate to achieve tasks that cannot be expressed as a set of pre-planned motions.

Multi-robot systems have begun to refer primarily to groups of mobile robots, which work together (or at least have a significant degree of interaction) when accomplishing a task. One of the first proposed applications was autonomous vehicle platooning, in which a group of autonomous vehicles follow a leader vehicle in single file. This has been shown to be a more efficient method of traffic flow [61], increase fuel economy [62] and decouples the complex decision and navigation aspect of driving from the simpler lateral and longitudinal control [63]. This has been studied for large-scale commuter and freight applications [64],



(a) Multi-robot grasping [60]

(b) Multi-robot platooning [67]

Figure 3.1 – Examples of applied multi-robot systems

but also for urban car-share or valet programs, and campus or airport shuttles [65], [66].

While the two examples just discussed are directly well established and directly applicable to everyday life, the past twenty years have seen intensive interest in multi-robot systems for complex, unplanned tasks. These are already becoming industry-ready, particularly in warehouse and logistics applications where fleets of mobile robots are used to transport goods, proving to be more versatile than previous intra-warehouse transportation methods. In many other aspect however, multi-robot systems are only beginning to extend beyond academia. Task such as collaborative object manipulation and transportation have been intensively studied for groups of ground robots [68]–[70], aerial robots [33], [71]–[73], and both [74], [75], but have not yet been widely adopted for real-world applications.

The object of this thesis, UAV fleets, is one of these multi-robot systems that may be on the verge of breaking into everyday use. Individual UAVs have, in the past five years, established themselves into commercial and government operations with consistently increasing use (see chapter 2). In the vast majority of cases, UAVs are used to perform perception tasks, and are simply required to carry a sensor. It stands to reason that for many tasks for which speed, guaranteed performance or multi-sensor data are required, the solution is to use multiple UAVs. There are many case studies where multiple UAVs may be beneficial, generally related to tasks such as area patrolling [76], disaster response [77], search and rescue [78], sensing, and inspection.

In [79], it is shown how multiple UAVs are able to establish a secure zone for which can be guaranteed to contain no hostile agents. Other works such as [80], [81] deal with the patrolling potential of multi-robot systems. In many of these security cases it is desirable to have a guaranteed result which a group of small robots may be able to achieve while a single larger and faster robot could not, furthermore the inherent redundancy of multi-robot

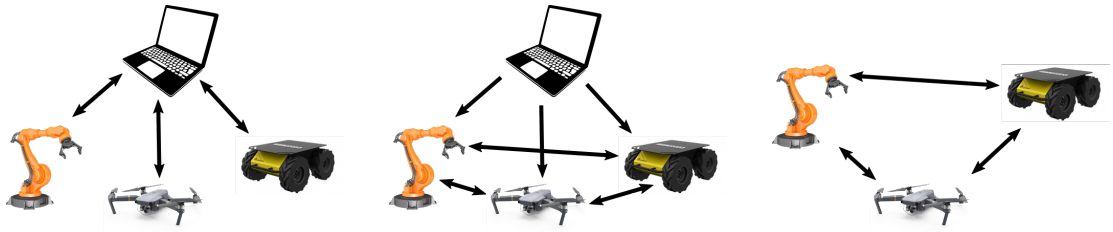
systems make them an attractive solution. Some applications of multi-robot systems have indeed arisen from the widespread availability of UAVs and the security threat they impose. Many works [82]–[84] deal with the encircling of an intruder by multiple UAVs in order to guarantee continued surveillance despite evasive manoeuvres.

Exploration, mapping, and scene reconstruction also benefit from being performed by multiple robots, achieve faster completion times [85], [86]. A similar conclusion is reached for using fleets of UAVs for assessing damage and risk scenarios during flooding [87] or other large footprint disaster zone, where UAVs can also serve as relays to replace damaged communication infrastructure [88]. Of course it is not sufficient to simply turn multiple robots loose into an environment and receive optimal results. How the robots behave collectively has been shown to drastically change the coverage time in multi-UAV search and rescue simulations [89], as well as in surveying ship hulls for damage using underwater robots [90]. In some cases, it is not only the task completion time that requires precise inter-robot coordination, but also the task quality as found when analysing the 3D reconstruction accuracy from images taken by fleets of UAVs with different formation structures [91]. This may have interesting implications with the increased use of UAVs in large scale infrastructure inspection and surveying, as the use of carefully coordinated groups of UAVs could both decrease the required time and increase the resulting precision compared to a less coordinated operation by the same equipment.

This section has hopefully provided the reader with an appreciation for the potential for multi-robot systems, and the following section will show the different multi-robot control architectures and how they are suited to various applications.

3.1.2 System Architectures

In traditional robots, there is generally a clear relationship between the user, controller, sensing and actuation abstraction layers. In a general case, the user (either a human or a program with some knowledge of the system and the desired actions) provides a task to the controller, the controller receives feedback from the sensors, determines an appropriate actuation command, and reports back to the user. In multi-robot systems however, the flow of information is more nuanced although it can often be classified based on the system architecture. For a system in which high speed communication, computing, and localisation can be assured, a similar flow of information as with a single robot may be employed.



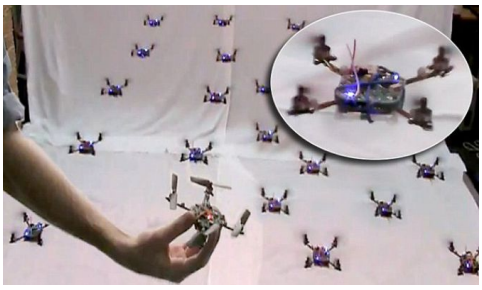
(a) Centralized (b) Decentralized (c) Autonomous

Figure 3.2 – Different architectures of multi-robot control

Each individual robot reports back to a common controller, which computes actuation for all robots based on the full state measurements of the system. This is known as centralized control and is represented in Fig. 3.2a. Many multi-robot systems including the previous aerial manipulation work at LS2N [3]–[5], [92], [93] make use of such controllers, as in laboratory settings the three critical features; communication, computing, and localisation are well controlled. In many situations however these three criteria may not be met and other control architectures must be considered.

If communication is insufficiently reliable, the system gains too many variables, or the robots are not able to share a common reference frame, decentralized control becomes important (as seen in Fig. 3.2b). In such a case the user will only supply high-level directives to the system, and each robot will decide its behaviour based on it's own knowledge, and potentially the communication from some or all of its peers. This means that the robots may interact and adapt to its local environment without intervention from the user, although of course high-level prompts from the user may be used to better guide the system. There may be some blurring of boundaries between centralized and decentralized control, however a good indicator is the notion of criticality between abstraction layers. In a centralized control system, cutting a communication channel to one robot will completely disable it. In the decentralized control architecture however, the robot isolated from the user will continue to function with some degree of autonomy, although maybe not as effectively. Another difference between the two control types is the scaleability of the system-wide control algorithm. At some point if sufficient robots are added to a centralized controller, the controller will not be able handle the scale of the control problem in real time. With decentralized controllers however, each agent computes its own control sub-problem and thus the scale of each of the sub-problems is limited and real-time behaviour is generally independent to the system size.

(a) A light show with hundreds of quadrotors flying in coordination [F7]



(b) A fleet of 20 small quadrotors [94]

Figure 3.3 – Large drone fleets controlled by centralized trajectory planners

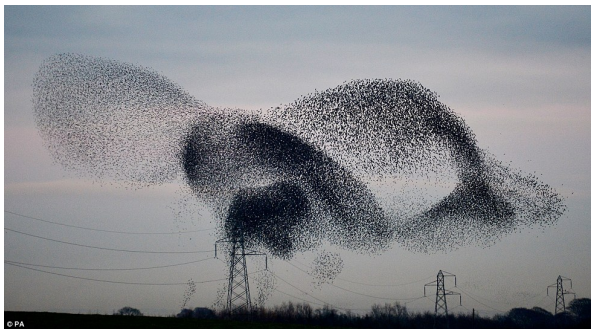
Going beyond decentralized control one may arrive at autonomous control (see Fig. 3.2c). In autonomous control, the user input is minimal, and it is the robots themselves determining the higher-level tasks that they will accomplish together. Like the boundary between centralized and decentralized control, there is also a boundary between decentralized and autonomous control as there is almost always human intervention somewhere. The main difference can be seen as the time criticality of user intervention, where a decentralized control may have a continuous input from the user, while an autonomous system would ideally be switched on, given a high level directive (or determine one for itself) and execute it without intervention or with occasional guidance.

3.2 Decentralization of Multi-UAV Systems

In this section we outline the challenges and benefits of moving from centralized to decentralized system architectures, as well as the primary methods and control strategies. We also discuss important tools in multi-robot control such as consensus algorithms.

3.2.1 Multi-UAV Navigation Strategies

For the moment there are few truly autonomous multi-robot systems, with most being limited to decentralized multi-robot systems where each robot has a defined task. Current autonomous multi-agent systems may include multi-UAV encirclement or pre-planned inspection routes, however a complete removal of continuous human intervention (and certainly supervision) moves towards the domain of artificial intelligence. There have however been many impressive demonstrations from Intel and other companies with over 2000 simultaneously flying UAVs equipped with LED lights, arranging themselves in



(a) A flock of birds [F8]

(b) A formation of birds [F9]

Figure 3.4 – Biological examples of flocking and formation flight

geometric patterns such as in Fig. 3.3a. These are highly controlled flights with preplanned trajectories and are not adaptable in real time to new configurations, loss of external localization, and other uncertainties [95]. The few recent online multi-UAV trajectory planners [96] rely on precise knowledge of the UAVs relative positions and thus they may thus be implemented only in highly controlled environments. In lab situations (see Fig. 3.3b) with motion capture systems (i.e. high-precision absolute localization) this is feasible [94] but even with a formation of 20 drones they are controlled in real time by dividing them into groups and controlling each group, reducing the complexity of the control problem. It is clear that in order to expand the capabilities of drone fleets in unmastered environments, decentralization is the key.

For some multi-robot systems such as a fleet of autonomous robots mapping an area [97], decentralization could simply augment each of the individual robots to 1) avoid collisions with nearby robots and 2) to not follow exactly the same path as nearby robots. This would result in many robots that behave in an individual manner with characteristics enabling them to coexist and be mutually more efficient. In many cases however as discussed in section 3.1.1, a collective behaviour is more desirable than a set of individual behaviours. In such a case, dedicated decentralized control schemes are necessary in order to achieve the desired effect. The implementation of these controllers depends on many factors, the most important of which are the kinematic constraints of the robot, the information available to each individual, and the nature of the desired motion.

The rest of this section outlines two of the most popular classes of decentralized fleet navigation strategies: flocking and formations. Both can be observed as biological phenomenon in groups of birds or fish, where each individual performs its own sensing, decision-making, and control, but together they move as a collective group. These two

navigation strategies have similarities in their implementation, but their objectives are very different as seen in Fig. 3.4. In flocking, the agents move as a fluid, unstructured body without a specific global shape, while formation control deals with maintaining a specified geometry. The application and implementation of these two types of decentralized behaviours are discussed hereafter.

3.2.2 Flocking Control

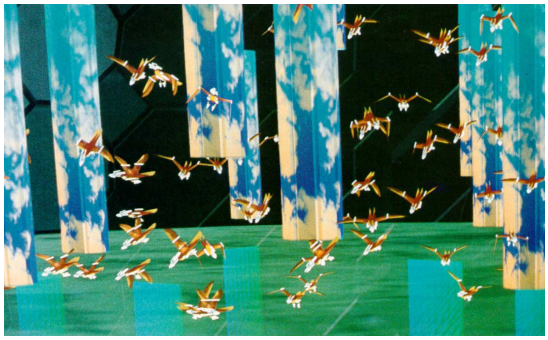
One of the first decentralized multi-robot controllers was developed not for robots but for computer agents. The initial work on flocking [98] was motivated by the development of computer simulations for studying bird flocks, and mentioned in passing the then-imaginary idea of creating computer generated extras in movies. The previous methodology for animating crowds was to specify a trajectory for each agent individually (i.e. a centralized method), a time consuming process with generally poor results when large numbers are required. The proposed flocking methodology was for each agent to behave independently, following three rules (known as Reynolds' rules):

- Avoid collisions
- Match the velocity of neighbouring agents
- Stay close to neighbouring agents

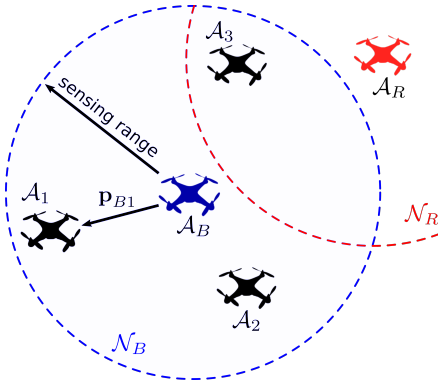
With these simple rules, it is possible to achieve natural crowd motion as seen by the simulation of an 80-bird flock (see Fig. 3.5a) in the initial paper published by Reynolds in 1987, and later for computer generated scenes or popular movies (mainly in battles involving thousands of individuals).

The concept of flocking was rapidly appropriated by the robotics domains, with the term *"Swarm Robotics"*[1] becoming broadly recognizable. There are of course many types of flocking algorithms, which are adapted to problems with different technological and environmental constraints. The many flavours of flocking algorithms are generally similar however in that each robot (or agent) $\mathcal{A}_i$ is able to detect some relative features of the robots in a bounded neighbourhood $\mathcal{N}_i$ around itself (and thus have a bounded computational

1. Note that while "swarm" may refer to small formations of robots, it is most often used to describe large unstructured groups of tens or hundreds performing flocking [99], thus we tend to avoid it in this manuscript except when explicitly referring to flocking.



(a) A swarm of computer generated birds, simulated in [98]



(b) A diagram of the local neighbourhood of flocking robots

Figure 3.5 – Flocking robots and their mutual interactions. In b) $\mathcal{A}_B$ is able to observe the three robots $\mathcal{A}_{1-3}$ within its neighbourhood $\mathcal{N}_B$, $\mathcal{A}_R$ is able to observe $\mathcal{A}_3$, but $\mathcal{A}_B$ and $\mathcal{A}_R$ do not observe each other. The other robots $\mathcal{A}_{1-3}$ may also sense those within their respective neighbourhoods.

complexity irrespective of the swarm size) as shown in Fig. 3.5b. $\mathcal{A}_i$ then computes its action so as to minimize an objective function with respect to the other agents in $\mathcal{N}_i$ [99].

Let us begin by considering the robots as simple double integrators on $\mathbb{R}^3$, where agents are controlled by their linear acceleration

$$\dot{\mathbf{p}} = \mathbf{v} \tag{3.1a}$$

$$\dot{\mathbf{v}} = \mathbf{u} \tag{3.1b}$$

where the control input is $\mathbf{u}$. Reynolds rules could be satisfied by finding an input that moves along the gradient of a potential function W to be optimized

$$\mathbf{u}_i = \sum_{j \in \mathcal{N}_i} k_p \nabla W \mathbf{p}_{ij} + k_v(\mathbf{v}_j - \mathbf{v}_i) \tag{3.2}$$

where $\mathbf{p}_{ij} = \mathbf{p}_j - \mathbf{p}_i$ is the position of $\mathcal{A}_j$ with respect to $\mathcal{A}_i$, and W is generally some function of the inter-agent distance $d_{ij} = ||\mathbf{p}_j - \mathbf{p}_i||$ which each agent $\mathcal{A}_i$ attempts to regulate. The effect of this control is that the difference in speed between each agent is minimized, as is the inter-robot distance objective $\sum_{j \in \mathcal{N}_i} d_{ij}^d - d_{ij}$, where d_{ij}^d is the desired inter-agent distance error. This will result in all agents converging in a finite time from an initial state to a state such where all measured inter-agent distances (i.e. all those belonging in some $\mathcal{N}$) are similar, and all agents move with the same velocity. The flock may be controlled by adding virtual agents $\mathcal{A}_v \in \mathcal{N}_i$ (or manually controller leaders) who are unaffected by the other

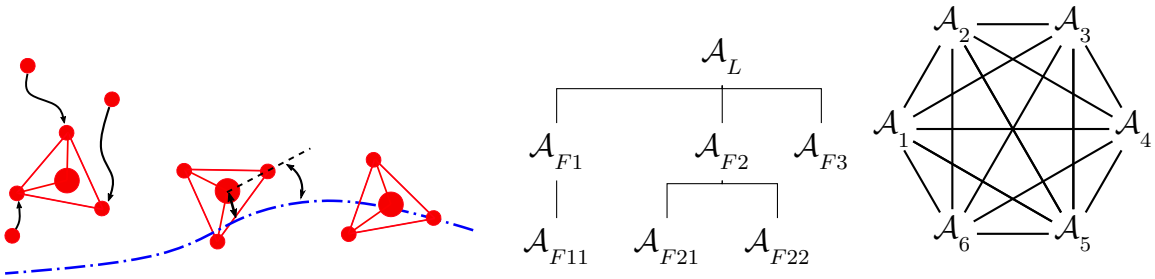
robots and whose velocity is observed by all robots, adding or subtracts average velocity from the flock when included in Eq. (3.2) [100].

There are many facets of study for flocking algorithms, to improve and to better understand their performance with robotics systems. Some works are interested in choosing objective functions with smooth gradient transitions, as the simple objective function $W = ||\mathbf{p}_j - \mathbf{p}_i||$ can cause poor convergence properties when applied to complex non-linear robot models (as opposed to single or double integrators) or at discrete control time steps [101]. Other considerations must be given to the non-holonomic constraints of the robots, as robots may be unable to perform the desired control (e.g. fixed-wing UAVs with a non-negligible turning radius), and thus the parameters of the flock such as inter-agent distance must be modified accordingly [102]. Sensing noise and delays can also result in inter-robot collisions, particularly at high speeds, and thus must be considered for fast-moving flocking [103]. Consideration must also be given to such factors as initial velocity conditions leading to the flock inadvertently splitting into multiple disconnected groups.

Other applied work in flocking robotics includes reducing the influence of adversarial or unplanned behaviour. In [104], algorithms are developed to identify robots that have either malfunctioned or maliciously infiltrated the flock of UAVs, and the flocking algorithms compensate for the undesired effect. Criteria are also developed to evaluate the "resilience" (or resistance to negative effects) of the system. A similar problem was approached in [105] using optimization to prevent the adversarial detection of a flock leader from the system motion. There is also current studies on restricted fields of view for flocking UAVs [106] and on the use of vision instead of distance in flocking algorithms [107]. While there is certainly much more to discuss regarding flocking, it is not the main topic of this dissertation and we will therefore move on to the second of the decentralized multi-robot behaviours; formation control.

3.2.3 Formation Control

Unlike flocking for which the primary objective is the aggregate motion of the robots, formation control aims to specify exact inter-agent geometries along with a collective displacement (see Fig. 3.6a). Formation control is a well known problem in biological systems, particularly in the avian sciences, where specified inter-agent geometries may serve to optimize energy consumption and protect from predators [108]. Inspired by this, recent

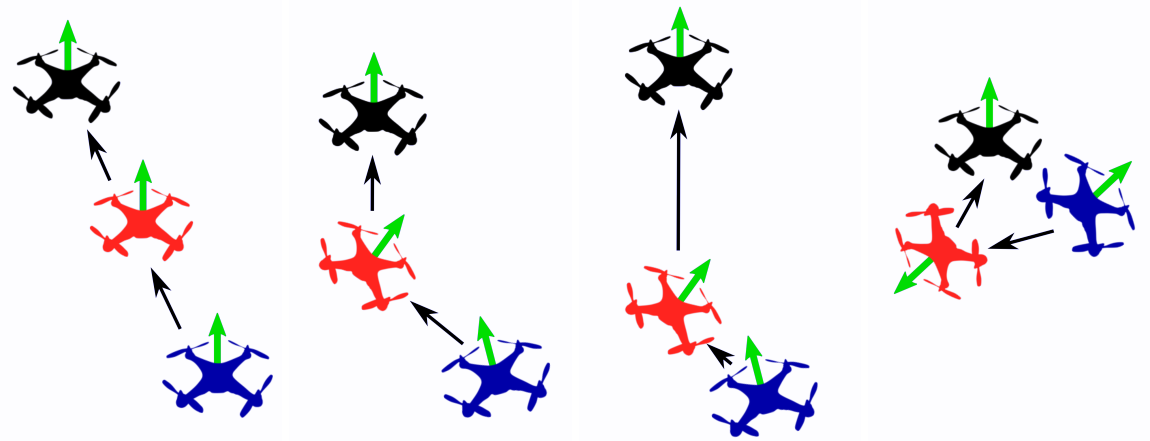


(a) Formation control tasks (b) Leader-Follower (c) General

Figure 3.6 – Decentralized formation control tasks and architectures, where agents joined by an edge are in one another's neighbourhood.

work in civilian aerospace showed that by flying long-haul aeroplanes in formations could potentially result in up to 24% reduction in climate impact and a 6% reduction in fuel cost [109]. For quadrotors which are often range limited by their energy storage capacity, this may also be leveraged to either increase the range of a group of UAVs by swapping positions in the formation, or to extend the range of a more important UAV by taking advantage of sacrificial UAVs for its aerodynamic assistance (similar to bicycle racing) [110]. Another task that may benefit from precise formations include distributed sensing, for instance to determine the direction of radio waves using small sensors on multiple UAVs as oppose to using a large antenna array [111]. Furthermore, there are numerous time and quality advantages to order formation control as discussed in section 3.1.1 which make the ability to fly in formation an attractive capability.

As with flocking, there are many ways to achieve formation flight, depending on the sensing, actuation, and computational limits of the agents. When aeroplanes fly in tight formation, each pilot is responsible for keeping a position relative to several specific neighbours instead of with all other aircraft in the formation, as the latter would be too complex [111]. Likewise, migratory birds also maintain a fixed position with respect to the one which precedes it [112]. Recalling the idea of neighbourhoods from the previous section, a similar concept is applied to formation control to limit the computational complexity of the controller. In the case of formation control however, the neighbourhood of one robot is a set of specific robots while in flocking it is the set of robots within a specific surrounding volume, and is therefore an effect of the control rather than a parameter. How the neighbourhood of each robot is defined is a key property of formations. In V-formations such is often seen in birds there is a hierarchical structure with a leader and followers, resulting in the tree-shaped architecture in Fig. 3.6b. On the other hand it may be preferable, or even necessary, to have

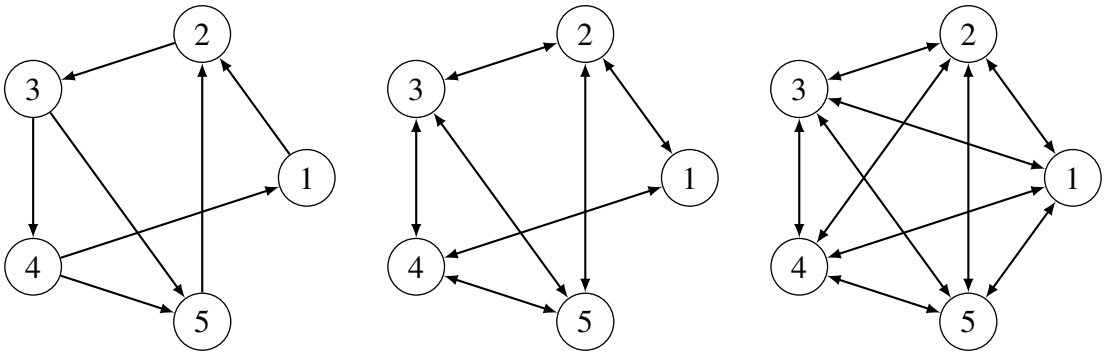


(a) Pose $\in SE(3)$ (b) Position $\in \mathbb{R}^3$ (c) Bearing $\in \mathbb{S}^2$ (d) Distance $\in \mathbb{R}^1$

Figure 3.7 – Formations with different types of inter-agent relative measurements. The desired formation is fully defined by inter-robot pose measurements in (a), but when partial measurements (b), (c), and (d) are used, the formation is not fully constrained. These partial measurements are all consistent with (a), but clearly they are insufficient to completely define the formation shape.

a more general leaderless formation architecture as shown in Fig. 3.6c. In either case, the connectivity between robots are externally specified and generally state invariant, although there is work dealing with adaptive connectivity to deal with occlusion [113].

The major factor in determining the architecture of a formation is the type of available measurements. If each robot is able to measure the position and orientation of another robot relative to itself, then the shape of the formation can be uniquely defined for hierarchical formation structures such as Fig. 3.6b. This may work well for humans over short distances (e.g. pilots flying in formation), however in many cases sensors are not able to detect the complete relative pose inter-robot necessary to define the formation geometry. In fact, relative pose measurement in unstructured environments remains difficult, although now artificial intelligence (AI) has begun to progress in that field. Looking at section 2.4.2 relating to relative sensing, it can be seen that there are generally three options: lidars measuring relative position (i.e. position but not orientation of $\mathfrak{F}_j$ in $\mathfrak{F}_i$), cameras measuring relative bearing, and range finders measuring distance. The problem with using low-information measurements is demonstrated in Fig. 3.7, for which a hierarchical formation of three drones is defined by relative pose, position, bearing, and distance measurements. It can be seen that without fully relative pose measurements, a purely hierarchical formation structure cannot be achieved. In such a case, more general architectures such as in Fig. 3.6 must be employed. This is discussed in more detail in the following section on graph theory.



(a) Directed graph (b) Bidirectional graph (c) Complete graph

Figure 3.8 – Some different types of graphs, where the label i corresponds to vertex $\mathcal{V}_i$.

3.3 Graph Theory for Formation Control

Graph theory plays a central role in multi-robot formation control. It allows a convenient mathematical expression of the manner in which the robots exchange information about the state of the formation. In this section we present the background of graph theory to make use of in the following sections.

3.3.1 Representation

Graph theory was developed by Euler in 1735 for his famous proof of the Konigsberg bridge problem, and is now fundamental for current applications such as machine learning, network analysis, electric and mechanical circuit analyses, logistics path planning, and even navigation apps. Developed to conveniently represent essential combinatorial information, A graph $\mathcal{G}(\boldsymbol{\mathcal{V}}, \boldsymbol{\mathcal{E}})$ may be defined as:

Definition 3.1 - Graph: *A mathematical structure composed of a set of vertices $\boldsymbol{\mathcal{V}} \in \mathbb{R}^{|\boldsymbol{\mathcal{V}}|}$ and a set of edges $\boldsymbol{\mathcal{E}} \in \mathbb{R}^{|\boldsymbol{\mathcal{E}}|}$ linking those vertices. Each vertex $\mathcal{V}_i \in \boldsymbol{\mathcal{V}}$ corresponds to some object (an event, a place, a robot), and each $\mathcal{E}_{ij} \in \boldsymbol{\mathcal{E}} = \mathcal{V}_i \times \mathcal{V}_j : \mathcal{V}_i, \mathcal{V}_j \in \boldsymbol{\mathcal{V}}$ indicates a link between $\mathcal{V}_i$ and $\mathcal{V}_j$.*

Graph theory is very much therefore a science of relationships, where each vertex (or node) is a subject, and each edge simply expresses some relationship between two subjects. This may take the form of an allowable displacement as in the Konigsberg bridge problem, an exchange of information as in the case of network analysis, or a measurement of state as is the case in formation control.

While graphs may be represented as lists, in this dissertation they are more commonly

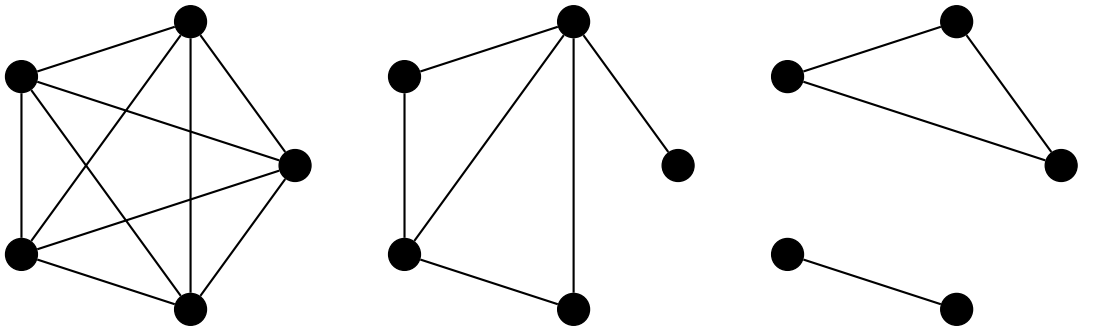
represented by diagrams, examples of which are shown in Fig. 3.8. Each vertex is represented by a circular node, while each edge is represented by an arrow leaving one vertex and entering another. If for two given vertices $\mathcal{V}_i$ and $\mathcal{V}_j$ there exists only a single edge $\mathcal{E}_{ij}$, this will be represented by an arrow leaving $\mathcal{V}_i$ and entering $\mathcal{V}_j$. Likewise, if the only edge joining the two is $\mathcal{E}_{ji}$, the arrow will leave $\mathcal{V}_j$ and enter $\mathcal{V}_i$. These are considered directed edges. If there exist $\mathcal{E}_{ij}, \mathcal{E}_{ji} \in \boldsymbol{\mathcal{E}}$, then it is represented by two superimposed arrows, and is considered as a bidirectional (or equally, as an undirected) graph edge. If the directionality is unimportant (e.g. all edges are bidirectional), the edges may be drawn without arrows. The term "neighbourhood" was already introduced in section 3.2.2, however we can generalized the concept to graph theory by saying that the neighbourhoods of a vertex consists of all the adjacent vertices

$$\mathcal{N}_i = \{\mathcal{V}_j\} \quad \forall j \text{ s.t. } \mathcal{E}_{ij} \in \boldsymbol{\mathcal{E}} \tag{3.3}$$

and thus flocking problem may be represented as a graph which changes as the robots move in and out of each others neighbourhoods.

3.3.2 Graph Properties

An important property of graphs is the completeness. A directed graph is considered to be complete (Fig. 3.9a) if every vertex is in the neighbourhood of every other vertex. A complete directed graph with $|\boldsymbol{\mathcal{V}}| = n$ vertices will therefore have $n(n-1)$ edges, as every vertex has $n-1$ edges leaving it towards every other vertex. Let us consider an algorithm decentralized across each node of the graph and that acts information related to each edge. Assuming each vertex has the same computational capability, the computational power of the graph is proportional to n, whereas the complexity of an algorithm dealing with all edges is proportional to $\mathcal{O}(n^2)$. The computational difficulty of such a problem is thus proportional to the number of vertices in the graph, and is therefore not highly scaleable. In many cases however the graph will not be complete, as in the case of flocking where each vertices neighbourhood is limited to the other robots within a given range. In this case there will be some finite upper limit to the size of the neighbourhood (let us call it u), and therefore the algorithmic complexity increases as $\mathcal{O}(un)$, while the computing increases proportional to n. In this case, so long as operations on the neighbourhood of any given vertex is computationally feasible, the algorithm may be scaled to a graph of arbitrary size.



(a) A complete graph (b) A connected graph (c) A disconnected graph

Figure 3.9 – Examples of different graph connectivities

A graph is considered as connected (Fig. 3.9b,c) if all vertices may be joined by traversing any combination of edges, and as disconnected otherwise.

It is often of interest to represent graphs in a matrix form. We begin by defining the adjacency matrix of a graph as $\mathbf{E}(\mathcal{G}) \in \mathbb{R}^{|\boldsymbol{\mathcal{V}}|\times|\boldsymbol{\mathcal{E}}|}$. Any given element e_{ij} of $\mathbf{E}$ is defined as

$$e_{ij}\begin{cases} 1 \text{ if the } j^{\text{th}} \text{ edge leaves } \mathcal{V}_i \\ -1 \text{ if the } j^{\text{th}} \text{ edge enters } \mathcal{V}_i \\ 0 \text{ otherwise} \end{cases} \tag{3.4}$$

and is sufficient to fully define all information contained by the graph. This is often used to create the graph Laplacian matrix $\mathbf{L}_{\mathcal{G}} = \mathbf{E}\mathbf{E}^T \in \mathbb{R}^{|\boldsymbol{\mathcal{V}}|\times|\boldsymbol{\mathcal{V}}|}$ which has important properties for the convergence of decentralized algorithms. Let us consider that $\mathcal{A}_i$ has some information x_i that must be matched with the information x_j of $\mathcal{A}_j$ (this could be states such as position and velocity, a desired state, the estimate of some property, etc...). If for each agent i, the information evolves as

$$\dot{x}_i = -\sum_{j\in\mathcal{N}_i} k\left(x_i - x_j\right) \tag{3.5}$$

where k is a positive scalar gain, then the Laplacian can express the evolution of the entire information $\mathbf{x} = [x_1 \dots x_{|\boldsymbol{\mathcal{V}}|}]$ across the graph as

$$\dot{\mathbf{x}} = -k\mathbf{L}_{\mathcal{G}}\mathbf{x} \in \mathbb{R}^{|\boldsymbol{\mathcal{V}}|\times 1} \tag{3.6}$$

and thus the convergence properties of the distributed system can be determined by the spectral properties of the Laplacian matrix. For a directed graph, the eigenvalues of $\mathbf{L}_{\mathcal{G}}$

in order of increasing value are necessarily $\boldsymbol{\lambda}_{\mathbf{L}} = [0 \ \lambda_2 \ \cdots \ \lambda_{|\boldsymbol{\mathcal{V}}|}]$. A necessary condition to guarantee the stability of Eq. (3.6) is that rank $(\mathbf{L}) = |\boldsymbol{\mathcal{V}}|$, thus requiring the second eigenvalue (or connectivity eigenvalue) to be $\lambda_2 > 0$ in order that the stability condition $|x_i - x_j| \rightarrow 0 \ \forall j \in \mathcal{N}_i$ as $t \rightarrow \infty$ be fulfilled. It has been shown that the connectivity eigenvalue is positive if there is at least one vertex with a directed path passing through all other vertices.

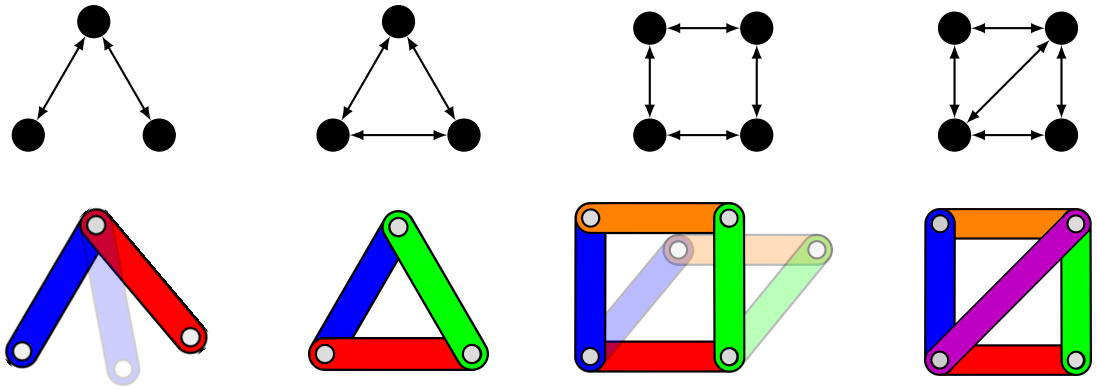
3.3.3 Frameworks and Rigidity

Up to this point, we have discussed how graphs have combinatorial and spectral properties, but little physical meaning. This changes with the idea of a framework $\mathcal{F}(\mathcal{G}, \mathbf{q})$, which associates an embedding vector $\mathbf{q} = [\mathbf{q}_1^T ... \mathbf{q}_{|\boldsymbol{\mathcal{V}}|}^T]^T \in \mathbb{R}^{|\boldsymbol{\mathcal{V}}|\dim(\mathcal{M})\times 1}$ to the graph, where $\mathbf{q}_i$ is the embedding of $\mathcal{V}_i$ on the manifold $\mathcal{M}$.

Definition 3.2 - Graph Embedding: *The association of a graph with a manifold $\mathcal{M}$ such that each vertex corresponds to a point on $\mathcal{M}$*

The embedding for a vertex of a multi-robot formation graph would of course be the state of the corresponding robot. Frameworks allow the extension of multi-robot system studies beyond the spectral space to a more meaningful vector-space by augmenting the combinatorial information of the graph with a geometric meaning. It also allows a graphical formulation of the concept of rigidity which is widely recognized in many engineering fields. Considering that each vertex $\mathcal{V}_i$ corresponds to a state $\mathbf{q}_i$, each edge $\mathcal{E}_{ij}$ represents a function $r(\mathbf{q}_i, \mathbf{q}_j) = r_{ij}$ constraining some aspect of $\mathbf{q}_i$ and $\mathbf{q}_j$. For an entire framework, there will be a relationship vector function $\mathbf{r}(\mathbf{q}) = [r_1 \cdots r_{|\boldsymbol{\mathcal{E}}|}]$, where r_i corresponds to the relationship function of the i^{th} graph edge in $\boldsymbol{\mathcal{E}}$. A framework is considered to be rigid if for any two sets of states $\mathbf{q}_1$ and $\mathbf{q}_2$ satisfying $\mathbf{r}(\mathbf{q}_1) = \mathbf{r}(\mathbf{q}_2)$, the frameworks $\mathcal{F}(\mathcal{G}, \mathbf{q}_1)$ and $\mathcal{F}(\mathcal{G}, \mathbf{q}_2)$ are congruent.

Rigidity can thus be considered as a property of a system that prevents it from deforming, most easily visualized on the $\mathbb{R}^2$ euclidean plane. We will provide an example of distance-based rigidity, which is the most well known as it is ubiquitous with static structure analysis, a core component of most early engineering curriculums. If one considers a mechanism of three pin-jointed links on the $\mathbb{R}^2$ plane, it is clear that there will be mobility between the links as is demonstrated in Fig. 3.10a. It is sufficient however to connect the free



(a) Flexible (b) Rigid (c) Flexible (d) Rigid

Figure 3.10 – A selection of graphs and their analogous distance rigidity mechanisms on $\mathbb{R}^2$

ends such that the mechanism forms a triangle (see Fig. 3.10b), and the resulting structure has no internal mobility. It can of course be displaced and turned around in space however there is no mobility between the individual links, and therefore the motion of one link will unambiguously determine the motion of all links. If we add a fourth link to make the square mechanism shown in Fig. 3.10c, we are all aware that rigidity will be lost, but by added a cross-brace as is done in Fig. 3.10d, rigidity will be recovered. If is important to be aware that this example is valid on the $\mathbb{R}^2$ manifold (i.e. $\mathbf{q} = [p_x \; p_y]$ considers only the planar positions of the vertices) with distance measurements (considering only the r function $||\mathbf{q}_j - \mathbf{q}_i|| = d_{ij}$) but not necessarily valid for an embedding on an arbitrary manifold and with an arbitrary measurement function $r(\mathbf{q})$. What we have just presented as "rigidity" is actually a property called "generic rigidity", and there are in fact other types of rigidity that exist, such as global rigidity and infinitesimal rigidity. Because what is presented here is sufficient however, we will not go deeper into the field.

An import concept in rigidity is the "Rigidity Matrix". It is the jacobean matrix

$$\mathbf{B} = \frac{\partial \mathbf{r}}{\partial \mathbf{q}} \in \mathbb{R}^{\dim(r)|\boldsymbol{\mathcal{E}}| \times \dim(\mathcal{M})|\boldsymbol{\mathcal{V}}|} \tag{3.7}$$

relating an infinitesimal change of the edge measurements an infinitesimal change of the embedding of the vertices. The rigidity matrix will change depending on the measurement function and embedding vector used, but generally will have a rank of rank $(B) \leq \dim(\mathcal{M})(|\boldsymbol{\mathcal{V}}| - 1)$. This implies there will be a kernel of $\text{rank}(\ker(\mathbf{B})) \geq \dim(\mathcal{M})$ spanning the nullspace of the rigidity matrix, for which all the embedding vector of all vertices may be moved in a coordinated manner so as to not affect the measured values between

them. Motions that are inside the kernel of any rigidity matrix are called trivial motions $\mathfrak{M}$, and in the case of the examples in Fig. 3.10 would result in a coordinated motion of the embedding vectors such that the rigid frameworks move in $SE(2)$ without changing shape. The inherently flexible frameworks have the same trivial motions, however $\ker(\mathbf{B})$ is increased in rank, and thus an extra DOF is added to the possible embedding motions [111].

3.4 Bearing Formation Control

Now that we have presented the motivation for studying MRS, have presented a number of different control classes and have given a background into graph theory, we are able to now present current methods related to the core of this dissertation on bearing formation control. Recalling the objective of formation control from section 3.2.3, bearing formation simply means that the desired and measured geometry is defined by inter-robot bearing measurements (see section 2.4.2.2 and appendix B for a reminder of the nature of bearings). In this section, we present state of the art on bearing formation control methods for quadrotors.

3.4.1 Robot Models

Each vertex of the graph of course represents a single robot, capable of some degree of sensing, actuation, and computing. If one considers a formation of generic robots that can arbitrarily move through space, the formation could be designed on the $SE(3)$ manifold, which is to say that the formation may be arbitrarily translated and rotated through space while preserving local measurements. The framework embedding for any agent in the formation could therefore be $\mathbf{q}_i = [\mathbf{p}_i^T \ \mathbf{q}_i^T]$. It was shown in the previous chapter that quadrotors have a coupling between their roll and pitch, and their translational dynamics. In fact, the flat output of a quadrotor contains only the position and yaw of the quadrotor. As such, it makes sense to define formation as moving on the $\mathbb{R}^3 \times \mathbb{S}^1$ manifold. Likewise for ground vehicles which do not have independently controllable roll, pitch, or vertical translations, the formation should be defined on the $SE(2)$ manifold.

Each robot (specifically each quadrotor-like UAV) in the formation can thrust be given a formation state $\mathbf{q}_i = [\mathbf{p}_i^T \ \psi_i]^T$. The robot of course still evolves in the $SE(3)$ space we all inhabit, and thus will have a roll θ_i and pitch ϕ_i, however this states are internal states

which are independent of the formation controller. Any vector $\mathbf{x}$ measured in $\mathfrak{F}_i$ can thus be projected onto the $\mathbb{R}^3 \times \mathbb{S}^1$ manifold by the equation

$$\mathbf{x}^{\mathbb{R}^3 \times \mathbb{S}^1} = \mathbf{R}_y(\theta_i)\mathbf{R}_x(\phi_i)\mathbf{x} \tag{3.8}$$

The previously described agent model describes how the static limitations of the agent are taken into account in the formation design, but they are also accounted for in the control loop. The vast majority of formation controllers use simple integrator models and trust to approximate the motion of agents. The first order agent model of a point is $\dot{\mathbf{p}} = \mathbf{v}$, and for a quadrotor on the $\mathbb{R}^3 \times \mathbb{S}^1$manifold it works out as

$$\begin{bmatrix} \dot{\mathbf{p}}_i \\ \dot{\psi}_i \end{bmatrix} = \begin{bmatrix} \mathbf{R}_z(\psi_i) & \mathbf{0} \\ \mathbf{0} & 1 \end{bmatrix} \begin{bmatrix} \mathbf{v}_i \\ \omega_{z,i} \end{bmatrix} \tag{3.9}$$

Controllers based on the first order model rely on generating a velocity trajectory for the quadrotor to follow, and assume that is is tracked relatively closely. Second order models are based on acceleration as discussed in the following chapter, and follow similar assumptions that the quadrotor can instantly produce the desired acceleration.

3.4.2 Formation Graph Model

Graphs are used to model the interaction between agents in a formation, as they are a convenient manner of expressing the combinatorial set of interactions between agents. When discussing decentralized control, there are two primary graph types; sensing graphs and communication graphs. Each of these graph types expresses each agent as a vertex in the graph, while information about one vertex accessible to another is represented by edges. A graph is thus simply a representation of a list of the information available to each agent.

If we assume $\mathcal{A}_i$ to be represented by the graph vertex $\mathcal{V}_i$ with no connected edges, the only information available to $\mathcal{A}_i$ is its own proprioceptive measurements. In our case we assume there is no absolute positioning or heading, thus $\mathcal{V}_i$ may only contain information about $\mathbf{v}_i \in \mathfrak{F}_i$, the roll and pitch, and inertial data from the IMU. If we then add a directed edge $\mathcal{E}_{ij}$ to the sensing graph, then along with its own intrinsic states, $\mathcal{A}_i$ also has access to some relative measurement of the state of $\mathcal{A}_j$ in $\mathfrak{F}_i$. In this manuscript, we deal exclusively with bearings, thus $\mathcal{A}_i$ has access the measurement $\boldsymbol{\beta}_{ij}$. The communication graph works

in the opposite way of the sensing graph, in that while the edge $\mathcal{E}_{ij}$ of the sensing graph makes information about $\mathcal{A}_j$ available to $\mathcal{V}_i$, the edge $\mathcal{E}_{ij}$ in the communication graph sends information about $\mathcal{A}_i$ to $\mathcal{V}_j$. Although intuitive, we emphasise that only information known to $\mathcal{V}_i$ may be sent to along an edge $\mathcal{E}_{ij} \forall j$, thus the states sent (e.g. $\mathbf{v}_i$) are necessarily also expressed in $\mathfrak{F}_i$.

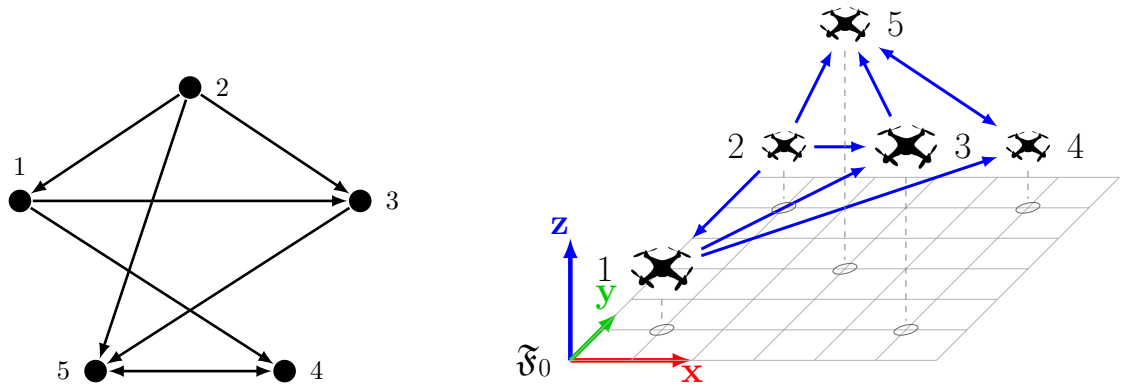
While the sensing and communication graphs both represent information flows, they are fundamentally different from a technological viewpoint. If we assume that a bearing sensor has a limited FoV, for a large formation it makes sense that each vertex will only have a set of edges

$$\boldsymbol{\mathcal{E}}_i = \{\mathcal{E}_{ij}\} \quad \forall j \in \mathcal{N}_i \tag{3.10}$$

that does not connect it to all other vertices, as some are outside the FoV of $\mathcal{A}_i$'s sensors. On the communications side, low power radio transmitters and receivers such as are found on small UAVs generally have an unlimited field of view, and thus are distance rather than direction dependant. Therefore when limited to a small geographic footprint, there is no issue with each agent establishing communication with every other agent forming a bidirectional complete graph for which each vertex has $|\boldsymbol{\mathcal{V}}| - 1$ bi-directed edges. This however is undesirable, as it cannot be scaled up indefinitely. If each agent is simultaneously transmitting and receiving data from every other agent, at some point the radio or wifi will become a bottleneck and communication will experience significant delays. Furthermore, any obstruction to the line of sight between two agents may degrade the quality of information exchange (depending of course on the equipment and the nature of the environment). It is therefore considered good practice to have a communication graph that is similar in dimension to the sensing graph, preventing scaling issues. In fact, many works consider that the communication graph is simply an un-directed sensor graph, and thus any edge $\mathcal{E}_{ij}$ is necessary and sufficient to imply the existence of $\mathcal{E}_{ij}$ and $\mathcal{E}_{ji}$ in the communications graph.

3.4.3 Bearing Formation Control

In the case of bearing formation control specifically, the goal is to control the geometry of the formation, and to manoeuvre it using its trivial motions. We start here with the first task, and move on to the later afterward.



(a) A directed graph, showing the flow of information between nodes

(b) An embedding of (a) in $\mathbb{R}^3 \times \mathbb{S}^1$, where each node i has an associated state $\mathbf{q}_i = [\mathbf{p}_i \in \mathbb{R}^3, \psi_i \in \mathbb{S}^1]$

Figure 3.11 – A directed formation graph and a possible framework

3.4.3.1 Formation shape control

Shape control consists of aligning the measured features with the desired measurements expressed in the moving robot frame $\mathfrak{F}_i$. If the formation is framework is rigid, than the shape of the formation will be fully defined by the measured feature. There are two types of bearing control: bearing-based control and bearing-only control [114]. In the case of the former, the robots may have relative position measurements, and inter-robot bearings are simply used to define the relative inter-robot geometry. This corresponds to 3D (or position-based) visual servoing commonly used in robotics.

Definition 3.3 - Bearing-based formation control: *A control for which the objective is to steer the robots to a relative geometry defined by a given set of inter-robot bearings.*

In bearing-only formation control however, the formation is not only defined by inter-robot bearings, but only have access to their relative bearings as well. This corresponds to 2D (or image-based) visual servoing commonly used in robotics.

Definition 3.4 - Bearing-only formation control: *A control for which the objective is to steer the robots to a relative geometry using only inter-robot bearing measurements.*

In this thesis we focus on the later as it is particularly suited to vision-based control [114] where depth is only very coarsely estimated, whereas the former requires relative position measurements which implies increased cost and weight. We nonetheless use the more general term of bearing-based in this thesis as one could still apply the methods bearing-only methods well if relative positions were measured, as we use bearing-only

control to accomplish bearing-based control tasks.

Let us assume for now that $\mathcal{A}_i$ is able to measure the relative bearings $\boldsymbol{\beta}_i = [\boldsymbol{\beta}_{i1}^T \cdots \boldsymbol{\beta}_{in}^T]^T \in \mathbb{S}^{2n}$ of n neighbouring agents at positions $\mathbf{p}_1$ to $\mathbf{p}_n$ in $\mathfrak{F}_0$. For now we will fix these other agents so that they are static in $\mathfrak{F}_0$, thus the only change in the measured bearings is due to the motion of $\mathcal{A}_i$. This corresponds to the classical eye-in-hand visual servoing used in robotics. The visual servoing task is then to find some input $\mathbf{u}_i$ for $\mathcal{A}_i$ that will reduce the error

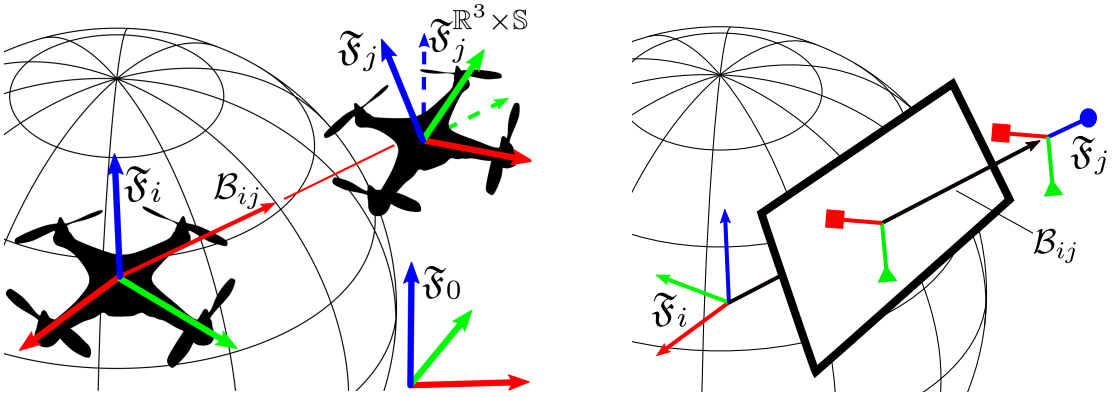
$$\mathbf{e}_{\boldsymbol{\beta} i} = \boldsymbol{\beta}_i^d - \boldsymbol{\beta}_i \tag{3.11}$$

between the measured and some desired set of bearings $\boldsymbol{\beta}_i^d$. This set of desired bearing may be chosen directly in bearing space, or may be reconstructed from some desired reference formation by Eq. (2.30). Care must be taken however, as multirotors are underactuated systems. Rotations around the $\mathbf{x}_i$ or $\mathbf{y}_i$ may be able to temporarily reduce the bearing error, but place the UAV in a dynamically unstable configuration which will result in longer-term divergence of the bearing error in addition to large internal dynamics. Some work has used approaches in $SE(3)$ such as the virtual spring approach [115] to permit some roll and pitch rotations with a virtual angular potential function included in the control law to maintain an equilibrium between stable feature errors and dynamically stable drone states. In multi-agent control however, it is more common to simply consider that the quadrotor is only able to act on its flat outputs. We therefore use the flat output model of the UAV $\mathbf{q}_i = [\mathbf{p}_i^T \ \psi_i]^T$ to account for the underactuation of the system, as any measured value in $SE(3)$ may be reprojected onto the flat model in $\mathbb{R}^3 \times \mathbb{S}^1$ by Eq. (3.8) and we may thus correct for the effect of roll and pitch motions on the visual system [116].

The first order visual servoing model relates the twist (i.e. body-frame linear and angular velocity) of the bearing sensor to the rate of change of the bearing feature. The flat twist of $\mathfrak{F}_i$ on the $\mathbb{R}^3 \times \mathbb{S}^1$ manifold expressed in $\mathfrak{F}_i$ is related to the rate of change of a single bearing $\boldsymbol{\beta}_{ij}$ by

$$\dot{\boldsymbol{\beta}}_{ij} = \mathbf{L}_{ij} \begin{bmatrix} \dot{\mathbf{p}}_i \\ \dot{\psi}_i \end{bmatrix} \qquad \text{where } \mathbf{L}_{ij} = \frac{\partial \boldsymbol{\beta}_{ij}}{\partial \mathbf{q}_i} \in \mathbb{R}^{3\times 4} \tag{3.12}$$

where $\mathbf{L}_{ij}$ is the bearing interaction matrix. This matrix may be calculated by the algebraic



(a) The bearing measurement $\boldsymbol{\beta}_{ij}$ of $\mathcal{A}_j$ by $\mathcal{A}_i$ (b) The action of the orthogonal projector $\mathbf{P}_{ij}$

Figure 3.12 – The bearing measurement of $\mathcal{A}_j$ relative to $\mathcal{A}_i$ (a) and the action of the bearings orthogonal projector (b)

partial differentiation of Eq. (2.30), and has been shown in [117] to be

$$\mathbf{L}_{i1} = \begin{bmatrix} \dfrac{-\mathbf{P}_{ij}}{d_{ij}} & -[\mathbf{z}_0]_\times \boldsymbol{\beta}_{ij} \end{bmatrix} \tag{3.13}$$

where $\mathbf{P}_{ij} = \boldsymbol{\mathcal{I}}_3 - \boldsymbol{\beta}_{ij}\boldsymbol{\beta}_{ij}^T$ is the orthogonal projection matrix for which the operation $\mathbf{P}_{ij}\mathbf{x}$ projects any vector $\mathbf{x} \in \mathbb{R}^3$ onto the $\mathbb{R}^2$ plane orthogonal to $\boldsymbol{\beta}_{ij}$ as in Fig. 3.12. Knowing how the bearing will react to a control input of the drone, a proportional control law may be designed to act on the bearing error

$$\mathbf{u}_i = k_p \mathbf{L}_{ij}^{\dagger} \left(\boldsymbol{\beta}_{ij}^d - \boldsymbol{\beta}_{ij} \right) \tag{3.14}$$

where $\mathbf{L}_{ij}^{\dagger}$ is the Moore-Penrose pseudo-inverse of the interaction matrix and k_p is a positive control gain. This results in an exponential convergence of the bearing error on the $\mathbb{S}^2$ manifold by a motion of $\mathfrak{F}_i$ orthogonal to $\boldsymbol{\beta}_{i1}$ and a rotation around $\mathbf{z}_i$.

Returning to our problem of a multirotor with n distinct bearing measurements, the classical method is to stack the bearings and the individual interaction matrices such that the bearing features are related to the camera twist by the first order interaction model

$$\dot{\boldsymbol{\beta}}_i = \underbrace{\begin{bmatrix} \mathbf{L}_{i1} \\ \vdots \\ \mathbf{L}_{in} \end{bmatrix}}_{\mathbf{L}_i} \mathbf{u}_i \tag{3.15}$$

which describes how the bearings change as a function of the motion of the observer $\mathcal{A}_i$. A simple proportional control law such as

$$\mathbf{u}_i = k_p \mathbf{L}_i^{\dagger} \mathbf{e}_{\beta i} \tag{3.16}$$

may the be used calculate the input $\mathbf{u}_i = [\mathbf{v}_i \ \omega_{z,i}]$ to $\mathcal{A}_i$ that with reduce the norm of the bearing error and which over time will move the camera such that the bearing measurements align with the desired bearings.

While visual servoing formation control as an individual which attempts to align its sensor features to their desired values, rigidity controls such as [118] considers the reciprocal nature of any motion $\mathbf{u}_i$ will have, not just on the measurements by $\mathcal{A}_i$, but also all measurements of $\mathcal{A}_i$ by other agents. Considering the stacked vector of all bearings in the formation $\boldsymbol{\beta} = [\boldsymbol{\beta}_1^T \cdots \boldsymbol{\beta}_{|\boldsymbol{\mathcal{E}}|}^T]^T$ as an output and the stacked formation embedding vector $\mathbf{q} = [\mathbf{q}_1^T \cdots \mathbf{q}_{|\boldsymbol{\mathcal{V}}|}^T]^T$ as the state, the bearing rigidity matrix $\mathbf{B}$ relates the change in state (i.e. the motion of the vertices) to the change in output (i.e. the motion of the bearing measurements) as

$$\dot{\boldsymbol{\beta}} = \mathbf{B}\dot{\mathbf{q}} \quad \text{where } \mathbf{B} = \frac{\partial \boldsymbol{\beta}}{\partial \mathbf{q}} \in \mathbb{R}^{3|\boldsymbol{\mathcal{E}}| \times 4|\boldsymbol{\mathcal{V}}|} \tag{3.17}$$

For a given bearing $\boldsymbol{\beta}_{ij}$, the rigidity matrix may be calculated analytically by taking the partial derivative of the bearing expressed by Eq. (2.30) with respect to the state vector $\mathbf{q} = [\mathbf{q}_i^T \ \mathbf{q}_j^T]^T$. Let us consider the possible motions that may influence $\boldsymbol{\beta}_{ij}$; a translation and yaw rotation of $\mathcal{A}_i$ and a translation of $\mathcal{A}_j$. If we assume $\mathcal{A}_i$ to be stationary, the change in the bearing due to the change in state of $\mathcal{A}_j$ is simply the component of $\dot{\mathbf{p}}_j$ orthogonal to $\boldsymbol{\beta}_{ij}$ projected onto the unit sphere around $\mathfrak{F}_i$. This can be expressed as

$$\frac{\partial \boldsymbol{\beta}_{ij}}{\partial \mathbf{q}_j} = \frac{\mathbf{P}_{ij} \mathbf{R}_i^T}{d_{ij}} \tag{3.18}$$

It can also be reasoned (or shown by differentiation) that a translational motion of $\mathcal{A}_i$ produces the negative effect of the motion of $\mathcal{A}_j$. Finally the effect of the yaw rate of $\mathcal{A}_i$ on the bearing can be found simply by taking the cross product $-\dot{\psi}[\mathbf{z}_0]_\times \boldsymbol{\beta}_{ij}$. The rows of the bearing rigidity matrix corresponding to $\frac{\partial \boldsymbol{\beta}_{ij}}{\partial \mathbf{q}}$ for an entire formation may therefore be

expressed as

$$\frac{\partial \boldsymbol{\beta}_{ij}}{\partial \mathbf{q}} = \begin{bmatrix} \mathbf{0} & \underbrace{\frac{\partial \boldsymbol{\beta}_{ij}}{\partial \mathbf{p}_i}}_{-\frac{\mathbf{P}_{ij}\mathbf{R}_i^T}{d_{ij}}} & \underbrace{\frac{\partial \boldsymbol{\beta}_{ij}}{\partial \psi_i}}_{-[\mathbf{z}_0]_\times \boldsymbol{\beta}_{ij}} & \mathbf{0} & \underbrace{\frac{\partial \boldsymbol{\beta}_{ij}}{\partial \mathbf{p}_j}}_{\frac{\mathbf{P}_{ij}\mathbf{R}_i^T}{d_{ij}}} & \mathbf{0} \end{bmatrix} \tag{3.19}$$

which relates the motion of all agents expressed in some common inertial frame to the change in measurement $\boldsymbol{\beta}_{ij}$. This can be extrapolated to the bearing rigidity matrix by vertically stacking Eq. (3.20) such that it expresses the effect of all agent motions on all agent bearings. Because however a yaw expressed in a common frame for all agents of the formation is necessarily possible, each row of the rigidity matrix may then be expressed rather in terms of the velocity in the frame of $\mathcal{A}_i$ who is the one measuring $\boldsymbol{\beta}_{ij}$. Expressed in the local frame the corresponding rows of the bearing rigidity matrix then become

$$\frac{\partial \boldsymbol{\beta}_{ij}}{\partial \mathbf{q}} = \begin{bmatrix} \mathbf{0} & -\frac{\mathbf{P}_{ij}}{d_{ij}} & -[\mathbf{z}_0]_\times \boldsymbol{\beta}_{ij} & \mathbf{0} & \frac{\mathbf{P}_{ij}\,{}^i\mathbf{R}_j^T}{d_{ij}} & \mathbf{0} \end{bmatrix} \tag{3.20}$$

With an analytic expression for the first order kinematics of the formation, it becomes possible to design a controller that selects an appropriate input for each agent to drive the output $\boldsymbol{\beta}$ to the desired value $\boldsymbol{\beta}^d$. The two main methods used in robotics for such a problem are the pseudo-inverse method and the jacobian transpose method. The former calculates the least-norm agent velocities as

$$\mathbf{u} = k_p \mathbf{B}^\dagger \mathbf{e}_\beta \tag{3.21}$$

where $\mathbf{B}^\dagger$ is the Moore-Penrose pseudo-inverse of the bearing rigidity matrix. The pseudo-inverse is generally the state of the art method for robotics control, however it will be remarked that this is only feasible by a centralized controller that is aware of the states of all agents. The jacobian transpose method calculates the control input as

$$\mathbf{u} = k_p \mathbf{B}^T \boldsymbol{\beta}^d \tag{3.22}$$

which has a closed form expression for the control input of $\mathcal{A}_i$

$$\mathbf{v}_i = -k_p \sum_{i,j \in \boldsymbol{\mathcal{E}}} \mathbf{P}_{ij} \boldsymbol{\beta}_{ij}^d + k_p \sum_{i,j \in \boldsymbol{\mathcal{E}}} {}^i\mathbf{R}_j \mathbf{P}_{ji} \boldsymbol{\beta}_{ji}^d \tag{3.23}$$

$$\omega_{z,i} = k_p \sum_{i,j \in \boldsymbol{\mathcal{E}}} \boldsymbol{\beta}_{ij}^T [\mathbf{z}_0]_\times \boldsymbol{\beta}_{ij}^d \tag{3.24}$$

Table 3.1 – Relationship between bearing error norm $||\mathbf{e}_{\beta}||$ and the equivalent angular error e_{α} between the measured and desired values of a single bearing measurement.

e_{α} (deg)	0	1	5	10	15	30	45	90	120	135	180
$||\mathbf{e}_{\beta}||$	0.0	0.02	0.09	0.17	0.26	0.52	0.77	1.41	1.73	1.85	2.0

which requires only the onboard measurements of $\mathcal{A}_i$, as well as the relative rotation and measurements between $\mathcal{A}_i$ and the agents which observe it (thus communication between neighbours is required). The relative rotation between agents can be achieved assuming that all agents are able to accurately measure and communicate their heading with respect to a common frame (by using a compass for example). Compasses however may experience significant perturbation from either external factors such as electrical fields or large metallic objects, or even from internal sources such as the quadrotors' motors [119]. These challenges may be overcome as described in section 3.4.4 using consensus algorithms.

When evaluating the success of a bearing formation controller, typically what is considered is the norm of the bearing error vector $||\mathbf{e}_{\beta}||$. Because we will be dealing with formations having different numbers of edges, in this paper we we consider a normalized version such that different formations may be compared with a common baseline. The bearing error norm may be expressed as

$$||\mathbf{e}_{\beta}|| = \sqrt{\mathbf{e}_{\beta 1}^T \mathbf{e}_{\beta 1} + \cdots + \mathbf{e}_{\beta|\boldsymbol{\mathcal{E}}|}^T \mathbf{e}_{\beta|\boldsymbol{\mathcal{E}}|}} \tag{3.25}$$

but we compare the results by the mean error norm $\frac{||\mathbf{e}_{\beta}||}{\sqrt{|\boldsymbol{\mathcal{E}}|}}$. Because each bearing is by definition a unit vector, the largest possible error for a given bearing is the diametrically opposite unit vector and therefore each bearing as well as the mean bearing error will be contained within the range $||\mathbf{e}_{\beta}|| \in [0, 2]$. Because readers will often be more familiar with angular representation of bearings, the corresponding angular error

$$e_{\alpha} = 2\sin^{-1}\left(\frac{||\mathbf{e}_{\beta}||}{2}\right) \tag{3.26}$$

is provided in table 3.1. The primary objective of formation control is to be able to manoeuvre the formation through space while the individual robots maintain the least possible magnitude of bearing error.

3.4.3.2 Bearing Formation Rigidity and Manoeuvring

We have then shown how the bearing formation controller may act upon the embedding so as to enforce the desired bearing measurements, which will define the shape of the formation if rigid. But we also know from section 3.3.3 that there will be a set of motions projected into the kernel of the bearing rigidity matrix which will not affect the measurement. It is proven that for formation control this set of trivial motions are

$$\mathfrak{M} \in \ker(\mathbf{B}) = \text{span}(\mathbf{v}_{\mathcal{F}}\ \dot{\psi}_{\mathcal{F}}\ \dot{s}_{\mathcal{F}}) \tag{3.27}$$

and consist of a translation velocity $\mathbf{v}_{\mathcal{F}} \in \mathbb{R}^3$, an angular velocity $\dot{\psi}_{\mathcal{F}}$, and a rate of scale change $\dot{s}_{\mathcal{F}}$ of the formation. If uncontrolled, the formation will move freely along its trivial motions, drifting in position, heading, and scale. This however can be used to control the formation, and it is proven in [118] that the kernel is such that any trivial motion may be mapped to a velocity and yaw rate of each agent. Thus if all robots receive a common trivial motion, they can each compute a command

$$\mathbf{v}_i^{\mathfrak{M}} = \mathbf{R}_i^T(\mathbf{v}_{\mathcal{F}} + \dot{\psi}_{\mathcal{F}}\mathbf{z}_0 \times (\mathbf{p}_i - \mathbf{c}_i) + \dot{s}_{\mathcal{F}}(\mathbf{p}_i - \mathbf{c}_i)) \tag{3.28a}$$

$$\dot{\psi}_i^{\mathfrak{M}} = \dot{\psi}_{\mathcal{F}} \tag{3.28b}$$

that may be applied by each without affecting the first task of the controller. Note that this requires all agents to share a common estimate for the position of the center of mass $\mathbf{c}_{\mathcal{F}}$ of the formation and an approximate formation scale. These may be estimated in a decentralized manner as discussed in the following subsection.

Because we can only control the trivial motions of the formation, we must ensure that the formation is rigid (i.e. there are no other motions beyond $\mathfrak{M}$that lie in $\ker(\mathbf{B})$).In the case of a generic rigid bearing framework, it is known that $\text{rank}(\mathbf{B}) = 4|\mathcal{V}| - 5$ and thus that $\text{rank}(\mathfrak{M}) = 5$. In [120] for distance formations embedding on $\mathbb{R}^3$, [121] for bearing formations embedded on $SE(2)$, and [118] for bearing formations on $\mathbb{R}^3 \times \mathbb{S}^1$, a semi-definite rigidity matrix $\mathbf{B}^T\mathbf{B}$ is define with the property that all $\dim(\mathcal{M})|\mathcal{V}|$ eigenvalues

$$\boldsymbol{\lambda}(\mathbf{B}) = [\lambda_1 ... \lambda_{\dim(\mathcal{M})|\mathcal{V}|}] = [0 \dots 0 \lambda_R \dots \lambda_{\dim(\mathcal{M})|\mathcal{V}|}] \tag{3.29}$$

are positive. For a rigid formation there would be $\text{rank}(\ker(\mathbf{B}))$ eigenvalues equal to zero

and thus in the case of quadrotor bearing formations the sixth smallest eigenvalue would be positive if the formation is rigid and zero in the case of a singular motion if rigidity is lost. This sixth smallest eigenvalue is then called the *"rigidity eigenvalue"* (and denoted by λ_R) as it gives a quantitative indicator of the rigidity of the formation. The generic rigidity of a formation may be evaluated by checking that $\lambda_R > 0$ for any random embedding, however as noted in [111] there are special sets of embedding for which rigidity is lost, and which are discussed in greater detail in part 3.

3.4.4 Decentralized Formation State Estimation

We have seen in the previous section that while we are able to control formations using relative bearing measurements, there are certain local and global states required which are not directly measurable. These states must then be estimated by a sharing of information between UAVs and through the use of consensus algorithms, which are note treated in detail by this thesis but are mentioned for completeness.

Consensus algorithms play an important role in multi-robot systems for the estimation of global properties by each of individual robots. Before the advent of multi-robot systems, consensus problems were studied extensively by the computer sciences community for distributed agents (that is different software instances on either the same or on separate computing hardware) that must work together to solve a problem. Per [122] "In the consensus problem a set of agents must all agree on a [binary] decision based on their initial states". This definition may still be relevant in a fully autonomous system of robots attempting to determine a task, but in general consensus problems have evolves to not just agreeing on a binary decision, but a set of non-discrete facts. Often in formation control, rather than having a leader robot who control the motion of the formation it is preferable to use a virtual leader [123] in order to increase system resilience. This virtual leader is effectively just a coordinate frame that all agents agree on, often located at the centroid of the formation. Let ξ be some global property of the system, for which each robot $\mathcal{A}_i$ will then have some estimate ξ. The objective of consensus algorithms is then to arrive at a state where $\xi_i - \xi_j = 0 \ \forall i, j$. Using consensus algorithms, it is then possible for each agent to estimate the non-available terms in the preceding sections, primarily a common formation heading even in the absence of reliable magnetometer data, and the center of the formation [7].

PART II

Dynamic Control of Formations

Abstract

This part details the contributions of this thesis pertaining to the improvement of the dynamic performance of decentralized bearing formation control. To begin with, we present a method based on second-order visual servoing that links the acceleration of the bearing features to an approximation of the quadrotors dynamic model on $\mathbb{R}^3 \times \mathbb{S}^1$. In validating the controller, numerous practical issues such as numerical conditioning and saturation handling are investigated. The second approach in this part is based on model predictive control (MPC) for visual servoing. An introduction to MPC and the justification for applying it to quadrotor bearing formation control is presented. We begin by developing the optimization problem for bearing formation control, and the closed loop MPC is show to have good convergence properties in simulation. The study is then extended to an experimental implementation, showing the benefit in terms of increased formation convergence dynamics and highly reactive steering. The MPC control is than augmented with a disturbance observer and various constraints such as collision avoidance for robust operation.

This part finishes with a chapter comparing the two controllers developed here to an existing single-integrator-based rigidity control, demonstrating that each controller has both benefits and drawbacks, with the newly developed controllers showing interesting dynamic properties.

List of Chapters

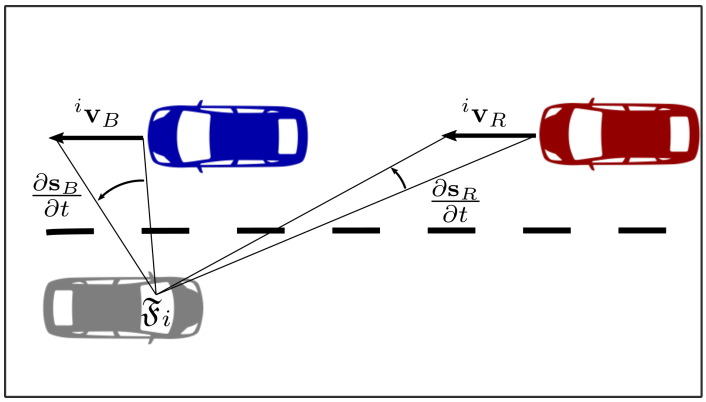
CHAPTER 4

SECOND ORDER BEARING FORMATION CONTROL

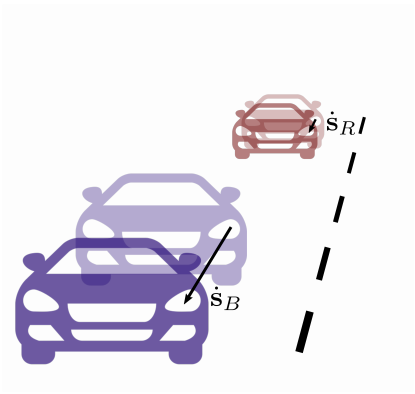
THE first method proposed in this thesis for increasing the dynamic performance of quadrotor bearing formations is to extend the first-order visual servoing approaches discussed in chapter 3 to a second order model. We start by presenting our motivation for attempting this control method, presenting the benefits of second order visual servoing (SOVS) in other applications. The SOVS model for quadrotors is then developed.

4.1 Background on SOVS

Visual servoing is a control method which, instead of fully reconstructing the robot state from sensor measurements, acts directly on the robot control to minimize the sensor feature error [124]. It typically relates the feature velocity $\dot{\mathbf{s}}$ to the camera velocity twist $\mathbf{t} = [\mathbf{v}^T \ \boldsymbol{\omega}^T]^T$, however in SOVS we relate the feature acceleration $\ddot{\mathbf{s}}$ to the camera twist derivative $\dot{\mathbf{t}}$, which may improve performance in high-dynamic applications. As quadrotors are controlled by thrust vectoring which relates directly to the body-frame acceleration, it is thought that this may provide a more natural link between the measurements and the



(a) A top view of an observer (gray) and two moving targets (red and blue).



(b) Two overlayed perspective views of the targets taken by the observer at a small time interval

Figure 4.1 – A perspective example of visual feature motion

quadrotor dynamics. In fact, the first order formation controller assumes that the quadrotor is able to track any velocity profile while neglecting the higher order terms in dynamic model. By linking the formation controller directly to the quadrotor acceleration, we may remove the unmodelled translational dynamics which should allow for a faster controller and direct access to the limiting actuation variable (the quadrotor thrust) for handling saturation. It has been furthermore shown in [125] that second order visual servoing results in smoother control signals and a lesser sensitivity to noise than classical visual servoing. Despite being shown in [116] that unmodelled translational dynamics can have significant adverse effects, few works explicitly consider the double-integrator translational dynamics of quadrotors in the context of bearing formation control. Those that have considered it (see [126] for a leader-follower bearing controller and [127] which uses static environmental features to control the formation nullspace) have been limited to numerical simulations with double integrator dynamics and special conditions beyond the general decentralized bearing formation architecture.

Moving beyond second-order translational robot dynamics, by differentiating the first order camera interaction model and considering only a stationary feature, the second order interaction model

$$\ddot{\mathbf{s}} = \mathbf{L}\dot{\mathbf{t}} + \dot{\mathbf{L}}\mathbf{t} = \mathbf{L}\dot{\mathbf{t}} + \mathbf{h}(\mathbf{s}, \mathbf{t}) \tag{4.1}$$

can be obtained [128], where $\dot{\mathbf{L}}$ is a hessian matrix which can be used to generate the vector $\mathbf{h}$ of terms that are non-linear with respect to the camera acceleration. Considering the simple case of a car accelerating from zero velocity in a straight line (as in Fig. 4.1), the driver will

notice the surrounding visual features accelerating, which relates to the car's acceleration by the interaction matrix $\mathbf{L}$. However when the car reaches the desired speed and stops accelerating, experience shows that the visual features continue to accelerate. For example, a car moving in the opposite direction will move slowly across the driver's field of view while it is distant, but very fast when observed through the side window. This visual feature acceleration dependant on the relative velocity is given by the quantity $\mathbf{h}(\mathbf{s}, \mathbf{t})$, and has been shown to be significant in manipulator-based visual servoing [129].

4.2 Development of SOVS Controller

In this section, we detail the design of a SOVS controller used for the formation control of quadrotors. We begin by developing the second order bearing model given the robots motion on its flat manifold, and then apply it to a feedback linearization control strategy.

4.2.1 Robot Model

We recall that in the previous chapters, it was shown how quadrotors are differentially flat, and furthermore that bearing formations controllers may be developed which consider only the first order kinematics of the flat outputs of the quadrotors. The robot model used in those controllers is thus

$$\dot{\mathbf{p}}_i = \mathbf{R}_z(\psi_i)\mathbf{v}_i \tag{4.2a}$$

$$\dot{\psi}_i = \omega_{z,i} \tag{4.2b}$$

where the states of each quadrotor are controlled by the input $\mathbf{u}_i = [\mathbf{v}_i^T \ \omega_{z,i}]^T$ consisting of the velocity and yaw rate expressed in $\mathfrak{F}_i$. The problem with this controller is that it assumes that quadrotors are able to closely track any velocity signal, while we know from the full model of the quadrotor presented in chapter 2 that this is a strong assumption outside of quasi-static hovering.

When a quadrotor is moving with a velocity $\mathbf{v}_A$ and we wish to achieve a new velocity $\mathbf{v}_B$, the quadrotor must first redirect it's attitude such that it is able to produce a thrust in the direction $\mathbf{v}_B - \mathbf{v}_A$. It then must exert a thrust in order to accelerate to this new velocity. We will assume for now the re-orientation component occurs very quickly compared to the

time taken to accelerate to the desired velocity (a weak assumption, as the quadrotor must re-orient prior to generating a correcting thrust). We can consider therefore the robot model

$$\dot{\mathbf{p}}_i = \mathbf{R}_z(\psi_i)\mathbf{v}_i \tag{4.3a}$$

$$\dot{\mathbf{v}}_i = [\mathbf{v}_i]_\times \dot{\psi}_i \mathbf{e}_3 + \mathbf{a}_i \tag{4.3b}$$

$$\dot{\psi}_i = \omega_{z,i} \tag{4.3c}$$

$$\dot{\omega}_{z,i} = \dot{\omega}_{z,i} \tag{4.3d}$$

where the robot inputs are $\mathbf{u}_i = [\mathbf{a}_i^T \ \dot{\omega}_{z,i}]^T$ for which $\mathbf{a}_i$ is the linear acceleration of the quadrotor in its flat frame $\mathfrak{F}_i$, and $\dot{\omega}_{z,i}$ is the angular acceleration of the quadrotor around $\mathbf{z}_i$. This second order model of the quadrotor on $\mathbb{R}^3 \times \mathbb{S}^1$ is a closer approximation of the full translational dynamic model in Eq. (2.17), thus we expect controllers developed using this model to have potentially higher performances.

4.2.2 Formation Controller

In order to make use of the higher-order robot model for formation control, the robot input must be related to the inter-robot bearing measurements. The first order bearing model between $\mathcal{A}_i$ and the observed agent $\mathcal{A}_j$ is shown in chapter 3 to be

$$\dot{\boldsymbol{\beta}}_{ij} = \mathbf{L}_{ij} \begin{bmatrix} \mathbf{v}_i \\ \dot{\psi}_i \end{bmatrix} + \frac{\mathbf{P}_{ij}{}^i\mathbf{R}_j}{d_{ij}} \mathbf{v}_j \tag{4.4}$$

where the first term relates the change in measurement to the observing robot's motion, and the second term relates the change in measurement to the motion of the observed robot expressed in $\mathfrak{F}_i$. The second order bearing model may then be found by taking the time derivative of Eq. (4.4).

$$\ddot{\boldsymbol{\beta}} = \underbrace{\mathbf{L}_{ij} \begin{bmatrix} \dot{\mathbf{v}}_i \\ \ddot{\psi}_i \end{bmatrix} + \dot{\mathbf{L}}_{ij} \begin{bmatrix} \mathbf{v}_i \\ \dot{\psi}_i \end{bmatrix}}_{\ddot{\boldsymbol{\beta}}_{ij|i}} + \underbrace{\frac{d}{dt}\frac{\mathbf{P}_{ij}{}^i\mathbf{R}_j}{d_{ij}} \mathbf{v}_j + \frac{\mathbf{P}_{ij}{}^i\mathbf{R}_j}{d_{ij}} \dot{\mathbf{v}}_j}_{\ddot{\boldsymbol{\beta}}_{ij|j}} \tag{4.5}$$

Which for the sake of convenience we group into terms that are linear with respect to the motion of $\mathcal{A}_i$ and term which are linear with respect to the motion of $\mathcal{A}_j$ (denoted as $\ddot{\boldsymbol{\beta}}_{ij|i}$

and $\ddot{\boldsymbol{\beta}}_{ij|j}$ respectively). This can then be arranged to algebraically isolate the control input $\mathbf{u}_i = [\mathbf{a}_i^T \ \dot{\omega}_{z,i}]^T$ in the form

$$\ddot{\boldsymbol{\beta}} = \mathbf{L}_{ij}\mathbf{u}_i + \mathbf{L}_{ij}\begin{bmatrix}[\mathbf{v}_i]_\times \dot{\psi}_i \mathbf{e}_3 \\ 0\end{bmatrix} + \dot{\mathbf{L}}_{ij}\begin{bmatrix}\mathbf{v}_i \\ \dot{\psi}_i\end{bmatrix} + \ddot{\boldsymbol{\beta}}_{ij|j} \tag{4.6}$$

This model therefore requires the calculation of the time derivative of the bearing interaction matrix $\mathbf{L}_{ij}$ which can be expressed as

$$\dot{\mathbf{L}}_{ij} = \frac{\mathrm{d}}{\mathrm{d}t}\begin{bmatrix}\dfrac{-\mathbf{P}_{ij}}{d_{ij}} & -[\mathbf{z}_0]_\times \boldsymbol{\beta}_{ij}\end{bmatrix} \tag{4.7}$$

which by applying the quotient rule (given that both $\boldsymbol{\beta}_{ij}$ and d_{ij} are time-variant), expands to

$$\dot{\mathbf{L}}_{ij} = \begin{bmatrix}\dfrac{-\dot{\mathbf{P}}_{ij}d_{ij} + \mathbf{P}_{ij}\dot{d}_{ij}}{d_{ij}^2} & -[\mathbf{z}_0]_\times \dot{\boldsymbol{\beta}}_{ij}\end{bmatrix} \tag{4.8}$$

The product rule must then be applied to calculate the derivative of the orthogonal projection matrix, which is

$$\dot{\mathbf{P}}_{ij} = \cancelto{0}{\dot{\mathcal{I}}_3} - \dot{\boldsymbol{\beta}}_{ij}\boldsymbol{\beta}_{ij}^T - \boldsymbol{\beta}_{ij}\dot{\boldsymbol{\beta}}_{ij}^T \tag{4.9}$$

where $\dot{\boldsymbol{\beta}}_{ij}$ can be calculated as a function of the $\boldsymbol{\beta}_{ij}$ and the relative agent velocities using the bearing rigidity matrix seen in section 3.4.3. While this could be obtained through the numerical differentiation of the bearing measurements, in practice this are too noisy (see appendix B) to give accurate results. The derivative of the distance must also be known but is easily calculated. As $\boldsymbol{\beta}_{ij}$ is the unit vector pointing to $\mathbf{p}_j$ expressed in $\mathfrak{F}_i$, only the component of $\mathbf{v}_i$ that is parallel to $\boldsymbol{\beta}_{ij}$ will result in an infinitesimal (negative) change in inter-agent distance. Likewise only the component of $\mathbf{v}_j$ expressed in $\mathfrak{F}_i$ parallel to $\boldsymbol{\beta}_{ij}$ will contribute to the (positive) change in inter-agent distance. We may therefore determine without resorting to calculus that

$$\dot{d}_{ij} = -\boldsymbol{\beta}_{ij}^T\mathbf{v}_i + \boldsymbol{\beta}_{ij}^T {}^i\mathbf{R}_j\mathbf{v}_j = -\boldsymbol{\beta}_{ij}^T(\mathbf{v}_i - {}^i\mathbf{R}_j\mathbf{v}_j) \tag{4.10}$$

and thus the inter-robot distance derivative relies solely on measured or (in the case of ${}^i\mathbf{R}_j$) well estimated values. We will then group all terms relating to the motion of $\mathcal{A}_i$ which are

non linear with respect to $\mathbf{u}_i$ in a single vector

$$\mathbf{h}_{ij|i} = \mathbf{L}_{ij} \begin{bmatrix} [\mathbf{v}_i]_\times \dot{\psi}_i \mathbf{e}_3 \\ 0 \end{bmatrix} + \dot{\mathbf{L}}_{ij} \begin{bmatrix} \mathbf{v}_i \\ \dot{\psi}_i \end{bmatrix} \tag{4.11}$$

which is a function of the current measured states of $\mathcal{A}_i$ and $\mathcal{A}_j$, along with an estimated inter-agent distance.

Having calculated all the terms required to estimate the bearing acceleration due to the motion of $\mathcal{A}_i$, it remains to determine the effect of the motion of $\mathcal{A}_j$, previously defined as $\ddot{\boldsymbol{\beta}}_{ij|j}$component in Eq. (4.6). This is very similar to the calculations previously shown, however one must account for the time-varying relative rotation between $\mathfrak{F}_i$ and $\mathfrak{F}_j$. These can be grouped as the vector

$$\mathbf{h}_{ij|j} = \frac{\left(\dot{\mathbf{P}}_{ij}{}^i\mathbf{R}_j + \mathbf{P}_{ij}{}^i\dot{\mathbf{R}}_j\right) d_{ij} - \mathbf{P}_{ij}{}^i\mathbf{R}_j \dot{d}_{ij}}{d_{ij}^2} \mathbf{v}_j - \mathbf{L}_{ij} \begin{bmatrix} [{}^i\mathbf{R}_j \mathbf{v}_j]_\times \dot{\psi}_j \mathbf{e}_3 \\ 0 \end{bmatrix} - \mathbf{L}_{ij} \begin{bmatrix} {}^i\mathbf{R}_j \mathbf{a}_j \\ 0 \end{bmatrix} \tag{4.12}$$

and which may be calculated using the same information as $\mathbf{h}_{ij|i}$. The full second order bearing model can thus be expressed as a linear function with respect to the control input of $\mathcal{A}_i$

$$\ddot{\boldsymbol{\beta}}_{ij} = \mathbf{L}_{ij}\mathbf{u}_i + \mathbf{h}_{ij|i} + \mathbf{h}_{ij|j} \tag{4.13}$$

and necessitates, along with the intrinsic measurements of $\mathcal{A}_i$, at least an estimate of the inter-agent distance, the linear velocity and acceleration of $\mathcal{A}_j$, and the relative yaw and yaw rates between $\mathcal{A}_i$ and $\mathcal{A}_j$. We furthermore combine the two constant terms as $\mathbf{h}_{ij} = \mathbf{h}_{ij|i} + \mathbf{h}_{ij|j}$ to formulate the model in the standard format $\ddot{\mathbf{x}} = \mathbf{M}\ddot{\mathbf{q}} + \mathbf{C}(\dot{\mathbf{q}}, \mathbf{q})$ of a second order robotic system. We remark that both components of the hessian are dependant on the inter-robot distance and some estimate of the inter-robot distance must therefore be used as a parameter. This may be taken from either a time-varying estimate as discussed in section 3.4.4, or simply set to the distance at the desired formation scale.

Using the visual servoing approach for formation control as well as the second order bearing model, a desired control signal must be chosen to minimize the bearing error $\mathbf{e}_{\boldsymbol{\beta}ij} = \boldsymbol{\beta}_{ij}^d - \boldsymbol{\beta}_{ij}$ where $\boldsymbol{\beta}_{ij}^d$ is the desired bearing. We choose the classical second order PD with feed-forward control law

$$\ddot{\boldsymbol{\beta}}_{ij} = k_p \mathbf{e}_{\boldsymbol{\beta}ij} + k_d \dot{\mathbf{e}}_{\boldsymbol{\beta}ij} + \ddot{\boldsymbol{\beta}}_{ij}^d \tag{4.14}$$

to force convergence of the bearing to the measured value. The desired control output may then be calculated using feedback linearization by

$$\mathbf{u}_i = \mathbf{L}_{ij}^{\dagger}\left(k_p\mathbf{e}_{\beta ij} + k_d\dot{\mathbf{e}}_{\beta ij} + \ddot{\boldsymbol{\beta}}_{ij}^{d} - \mathbf{h}_{ij}\right) \tag{4.15}$$

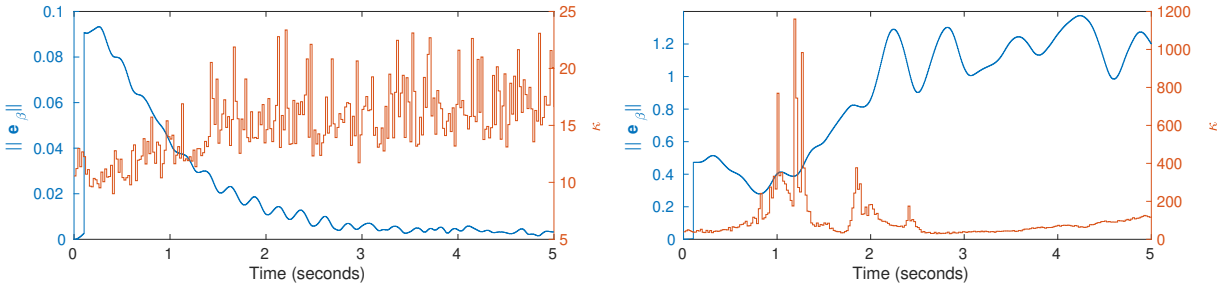
It is important to note that this is currently the control for an agent with a single bearing measurement. Recalling that $\mathbf{u}_i \in \mathbb{R}^3 \times \mathbb{S}^1$ and $\boldsymbol{\beta}_{ij} \in \mathbb{S}^2$, the problem is under constrained, and there are an infinite number of solutions to this control problem. As such, the pseudo-inverse operation returns the least norm actuation that will reduce the bearing error, however this lacks a physical significance because there is no direct relationship between the angular and linear accelerations which compose $\mathbf{u}_i$. In formation control however (as with visual servoing) there are often more than a single measured feature for a given agent. As presented in chapter 3, the solution is to stack the terms of each separate visual servoing model, thus the control law becomes

$$\underbrace{\begin{bmatrix}\mathbf{a}_i\\ \ddot{\psi}_i\end{bmatrix}}_{\mathbf{u}_i} = \underbrace{\begin{bmatrix}\mathbf{L}_{i1}\\ \vdots\\ \mathbf{L}_{in}\end{bmatrix}}_{\mathbf{L}_i^{\dagger}} \underbrace{\begin{bmatrix}k_p\mathbf{e}_{\beta i1} + k_d\dot{\mathbf{e}}_{\beta i1} + \ddot{\boldsymbol{\beta}}_{i1}^{d} - \mathbf{h}_{i1}\\ \vdots\\ k_p\mathbf{e}_{\beta in} + k_d\dot{\mathbf{e}}_{\beta in} + \ddot{\boldsymbol{\beta}}_{in}^{d} - \mathbf{h}_{in}\end{bmatrix}}_{\text{auxiliary control vector } \mathbf{w}_i} \tag{4.16}$$

and in the case of $n = 2$ is a fully constrained problem with a single solution[1]. If $n < 2$, the problem is under-constrained and we will seek to drive the error to zero with the least control input, while in the case of the over-constrained problem (where $n > 2$) the control output is such as to minimize the magnitude of the bearing error.

The second order control law is tested in simulation (in simulink, considering the full dynamic model of a quadrotor with limited propeller thrusts and rise times) for a basic bearing formation composed of three agents, where each agent observes the other two. Error is included on the bearing measurement, the inter-agent estimated distance, and the drone states. In the first simulation shown in Fig. 4.2a, the desired formation begins close to the initial formation, and we ensure that the formation is not near a singular configuration. It can be seen that the bearing error reduces to zero. Furthermore it is seen that there are no large

1. Except in the case of singularities



(a) Simulation starting with low initial error in a highly non-singular configuration

(b) Simulation starting with higher initial error and crossing a near-singular configuration

Figure 4.2 – Simulation results for a basic second order visual servoing control. The bearing error is in blue (left axis), and the largest conditioning number of the interaction matrices in red (right axis).

spikes in the conditioning number κ of the interaction matrices. The conditioning number of a matrix is the ratio between its largest and smallest eigenvalues, and is an indicator of numerical stability. In a poorly conditioned system (with a large conditioning number), a small change in the independent variable (state measurements) will have a large effect on the dependant variable (the control signal). In the second simulation shown in Fig. 4.2b, the desired formation is far from the initial formation, and the formation must pass near to a singular configuration. It is seen that initially (until around 1 s) there seems to be a convergence of the bearing error, but that the conditioning number of at least one of the interaction matrices becomes large and the system becomes unstable. This demonstrates the need to condition the system to be numerically stable near singular configurations. It is worth noting that the rigidity controller was also applied to these same two simulated conditions and performed well in both cases, as there is no matrix inversion. To demonstrate where the interaction matrix may have singularity problems, Fig. 4.3 shows the conditioning for sets of bearings. It can be seen that generally ill conditioning is an issue when there are two observations that are either co-linear or are such that the bearings lie on the surface of a common vertical cylinder centred on the observer.

4.2.3 Singularity Handling

There are several different techniques for dealing with poorly conditioned matrices found in existing robotics literature. The pseudo-inversion of the interaction matrix used to calculate to control signal is simply performing a Gauss-Newton optimization to calculate the control value which best forces the system in the desired direction. This may be formulated

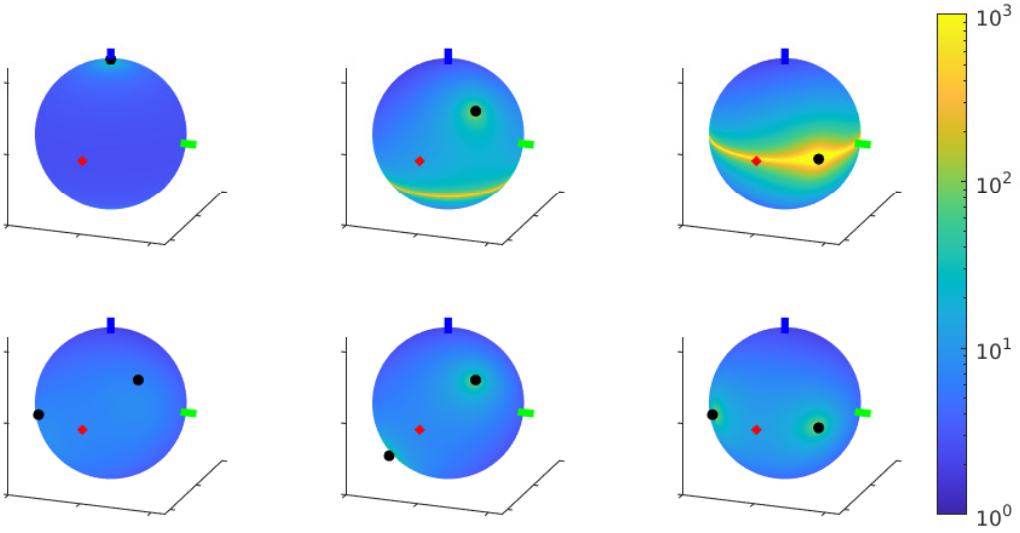


Figure 4.3 – Conditioning number of $\mathbf{L}_i$ (color) for sets of bearings. One (top row) or two (bottom row) bearings are fixed (represented by black points), and another bearing is tested over the whole surface of the unit sphere.

as a minimization problem

$$\min_{\mathbf{u}_i} ||\mathbf{L}_i\mathbf{u}_i - \mathbf{w}_i||^2 \tag{4.17}$$

where $\mathbf{w}_i$ is the auxiliary control vector from Eq. (4.16). When $\mathbf{L}_i$ loses rank (or as its smallest positive eigenvalue approaches zero), the desired motion requires a very large control output along the singular direction. Because of measurement noise and high numerical sensitivity near singularities, this can result in saturated chattering control signals which tend to cause instability in underactuated systems like quadrotors. Singular regularization methods may be used to modify the optimization problem so that the output is generally a similar direction to the original problem, but avoiding large motions in poorly conditioned directions. This can be done by "damped least-squares" for example [130], where the optimization is re-formulated as

$$\min_{\mathbf{u}_i} ||\mathbf{L}_i\mathbf{u}_i - \mathbf{w}_i||^2 + \gamma||\mathbf{u}_i||^2 \tag{4.18}$$

where $\gamma > 0$ is a constant damping gain. The singular values of $\mathbf{L}_i^\dagger$ then become $\frac{\sigma}{\sigma^2+\gamma} > 0$ where σ is the square root of the smallest positive eigenvalue $\mathbf{L}_i$. This however changes the formulation of the control problem, so that a sub-optimal convergence is performed everywhere. If the damping is too small, then numerical stability may not be guaranteed, while for large damping gains the system will converge slowly. A better solution would be to

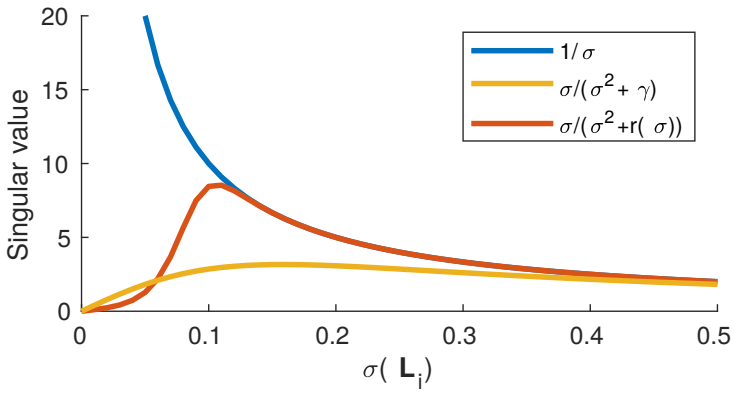


Figure 4.4 – Plot of the singular value with respect to the eigenvalue of the interaction matrix for the unregulated case (blue), regulated with least squares damping (yellow), and by SVD with bell support function (red)

selectively apply damping only in the singular direction and only when necessary to ensure stability.

The solution we chose is to apply a correcting factor using singular value decomposition (SVD), a method of factoring a matrix into its orthogonal and singular components [131], [132]. Given this factorization, we are able to modify the interaction model only where it is near to singularities. While any matrix may be factored using SVD, we are interested here in the factorization of the bearing interaction matrix as

$$\mathbf{L}_i = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T \tag{4.19}$$

where $\mathbf{\Sigma}$ contains a diagonal matrix of the square roots of the eigenvalues of both $\mathbf{L}_i\mathbf{L}_i^T$ and of $\mathbf{L}_i^T\mathbf{L}_i$ in decreasing order

$$\mathbf{\Sigma} = \begin{bmatrix} \text{diag}(\sigma_1 \dots \sigma_r) \\ \mathbf{0} \end{bmatrix} \qquad \text{where } r = \text{rank}\left(\mathbf{L}_i\right) \tag{4.20}$$

and as $\mathbf{L}_i$ approaches a singular condition, it loses rank, thus $\sigma_r \to 0$. The pseudo-inverse of a matrix may also be calculated from the SVD decomposition as

$$\mathbf{L}_i^\dagger = \mathbf{V}\mathbf{\Sigma}^\dagger\mathbf{U}^T \quad \text{where } \mathbf{\Sigma}^\dagger = \begin{bmatrix} \text{diag}(\frac{1}{\sigma_1} \dots \frac{1}{\sigma_r}) & \mathbf{0}^T \end{bmatrix} \tag{4.21}$$

and thus we can see the problem when any given eigenvalue approached zero. We therefore apply a method used to handle kinematic singularities in serial robotic manipulators [133],

[134], and design the least-square damping matrix to be a function of the singular values of the interaction model using a bell shaped function.

$$\mathbf{R} = \text{diag}\left(r_1(\sigma_1)\cdots r_r(\sigma_r)\right) \text{ where } r_i = he^{\frac{-\sigma_i^2}{2\mu^2}} \tag{4.22}$$

where h is the height of the bell curve and μ is it's standard deviation, or width. This leads to the optimization problem

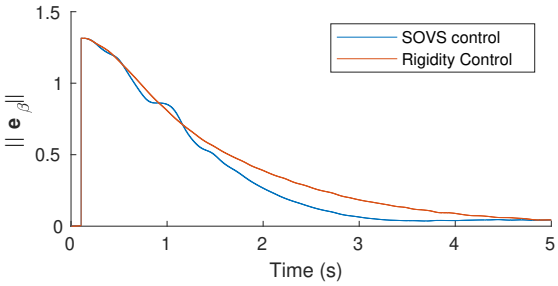
$$\min||\mathbf{L}_i\mathbf{u}_i - \mathbf{w}_i||^2 + ||\mathbf{R}^T\mathbf{u}||^2 \tag{4.23}$$

where $\mathbf{R}$ in only non-zero near singularities in the direction of the degenerate motion. The control input can therefore be calculated as a function of the factored and modified interaction matrix [134] as

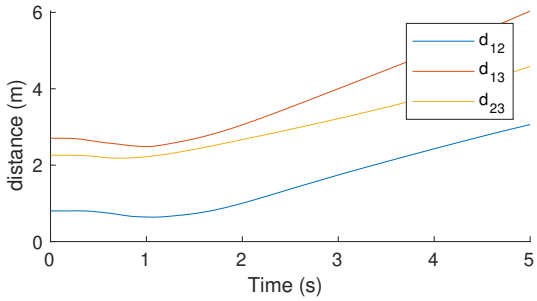
$$\mathbf{u}_i = \begin{bmatrix} \mathbf{a}_i^\beta \\ \dot{\omega}_{z,i}^\beta \end{bmatrix} = \mathbf{V}\left(\mathbf{\Sigma}^T\mathbf{\Sigma} + \mathbf{R}\right)^\dagger \mathbf{\Sigma}^T\mathbf{U}^T\left(k_p\mathbf{e}_{\beta i} + k_d\dot{\mathbf{e}}_{\beta i} + \ddot{\boldsymbol{\beta}}_i^d - \mathbf{h}_i\right) \tag{4.24}$$

where the singular values (i.e. the diagonal values of the $\left(\mathbf{\Sigma}^T\mathbf{\Sigma} + \mathbf{R}\right)^\dagger \mathbf{\Sigma}^T$ term) will be $\frac{\sigma_i}{\sigma_i^2 + r_i}$ (see [130], and will completely damp out motion along the singular direction when $\sigma_i \to 0$. The damping is therefore activated along the direction of motion for which the inverse is ill conditioned, while when the interaction matrix is non-singular, it is almost unaltered (see the red line in Fig. 4.4).

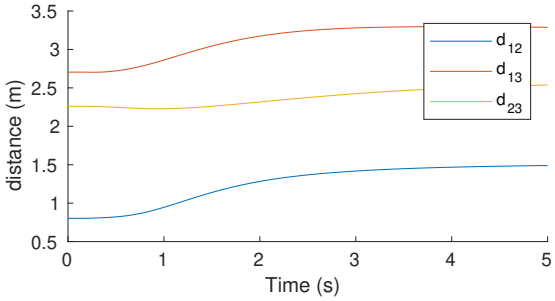
This method is very effective at ensuring the controller is numerically stable throughout the workspace of the formation, however it requires the tuning of the two regularization function variables, h and μ. Increasing the value of h will provide more aggressive damping (at the expense of increasingly suboptimal minimization of Eq. (4.17)), and increasing μ will increase the distance from singularities that this regularization begins to act. Note that here we have simply remarked upon the fact that singularities exist and provide a solution for maintaining numerical stability when near to singular configurations. This however is only a treatment of singularities in the interaction matrix, not the rigidity matrix, which has a difference effect on formation control and is studied in greater depth in chapter 8.



(a) Bearing error evolution for the formation using SOVS and Rigidity control methods



(b) Inter-agent distances with SOVS controller



(c) Inter-agent distances with Rigidity controller

Figure 4.5 – A demonstration of the first and second order open-loop dynamics in $\ker(\mathbf{B})$ for a 3-quadrotor formation having the same initial and final configurations.

4.2.4 Manoeuvring in the Formation Nullspace

While we have shown that a well-performing SOVS formation controller is able to effectively regulate the inter-agent bearings, we must equally control the motion of the formation. This is somewhat more complicated with the second order bearing model, because the quadrotors are controlled in acceleration instead of velocity. In the absence of an external steering command, when a velocity controlled formation reaches a null bearing error, the velocity of the agents becomes zero and the positions of each agent in the formation converges to constant values. In the acceleration controlled formation however, a null bearing error (and bearing error derivative) results in zero acceleration of the agents. This won't constrain velocity of the agents of the system, and in fact, in all but the most exceptional of cases, will tend to result in a formation undergoing constant translation, rotation, and expansion rates inside the nullspace of the bearing rigidity matrix. This is demonstrated in Fig. 4.5 with the bearing error converging for both controllers. After bearing convergence however, the SOVS controller results in the inter-agent distances increasing at a constant rate (indicating that the formation is expanding and thus the UAVs are moving with a constant non-zero speed), while for the first-order rigidity controller the inter-agent distances have stabilized to constant values.

Recalling that each quadrotor is able to measure their body-frame velocity and acceleration, we can therefore calculate an acceleration to regulate the trivial motions of the formation. Assuming that every agent in the formation received a coordinated signal $\mathbf{u}_{\mathfrak{M}} = [\mathbf{v}_{\mathcal{F}}\ \dot{\psi}_{\mathcal{F}}\ \dot{s}_{\mathcal{F}}]$ and is able to estimate its position $\mathbf{r}_i = \mathbf{p}_i - \mathbf{c}_{\mathcal{F}}$ and rotation $\mathbf{R}_i$ with respect to an arbitrary frame attached to the COM of the formation, each individual agent may express its desired velocity and yaw rate in $\mathfrak{F}_i$ as [118]

$$\mathbf{v}_i^{\mathfrak{M}} = \mathbf{R}_i^T(\mathbf{v}_{\mathcal{F}} + \dot{\psi}_{\mathcal{F}}\mathbf{z}_0 \times \mathbf{r}_i + \dot{s}_{\mathcal{F}}\mathbf{r}_i) \tag{4.25a}$$

$$\dot{\psi}_i^{\mathfrak{M}} = \dot{\psi}_{\mathcal{F}} \tag{4.25b}$$

This may then be used by each agent to calculate its acceleration required a reference trajectory of trivial motions, and in the event that the trajectory is generated smoothly, an acceleration feed-forward term can be added which will help to minimize tracking error as

$$\mathbf{a}_i^{\mathfrak{M}} = \mathbf{R}_i^T\mathbf{a}_{\mathcal{F}} + k_p(\mathbf{v}_i^{\mathfrak{M}} - \mathbf{v}_i) \tag{4.26a}$$

$$\dot{\omega}_{z,i}^{\mathfrak{M}} = \ddot{\psi}_{\mathcal{F}} + k_p(\dot{\psi}_i^{\mathfrak{M}} - \omega_{z,i}) \tag{4.26b}$$

where $\mathbf{a}_{\mathcal{F}} = \dot{\mathbf{v}}_{\mathcal{F}}$, however care should be taken to ensure a smooth desired acceleration of the nullspace velocity command. This is also insufficient however, as we know that points of set of rotating points will have a tendency to move outwards from the center of rotation, and thus we must add an artificial centripetal force to the controller to prevent the formation from expanding. Let us consider $\mathcal{A}_i$ to be a point on a rigid body at a displacement of $\mathbf{r}_i$ from the body's origin. If the body has an angular velocity $\boldsymbol{\omega}$ and angular acceleration $\boldsymbol{\alpha}$ about its origin, the linear acceleration of $\mathcal{A}_i$ can be described as

$$\mathbf{a}_i^{\mathfrak{M}} = \boldsymbol{\alpha} \times \mathbf{r}_i + \boldsymbol{\omega} \times (\boldsymbol{\omega} \times \mathbf{r}_i) \tag{4.27}$$

where the first term is the tangential acceleration of $\mathcal{A}_i$ and the second term is the so-called centripetal acceleration. As the only angular motion of the formation on the $\mathbb{R}^3 \times \mathbb{S}^1$ manifold is around $\mathbf{z}_0$, the manoeuvring acceleration can be calculated as

$$\mathbf{a}_i^{\mathfrak{M}} = \mathbf{R}_i^T\left(\mathbf{a}_{\mathcal{F}} + \ddot{\psi}_{\mathcal{F}}\mathbf{z}_0 \times \mathbf{r}_i - \dot{\psi}_X^2\mathbf{r}_i\right) + k_p\left(\mathbf{v}_i^{\mathfrak{M}} - \mathbf{v}_i\right) \tag{4.28a}$$

$$\dot{\omega}_{z,i}^{\mathfrak{M}} = \ddot{\psi}_{\mathcal{F}} + k_p\left(\dot{\psi}_i^{\mathfrak{M}} - \omega_{z,i}\right) \tag{4.28b}$$

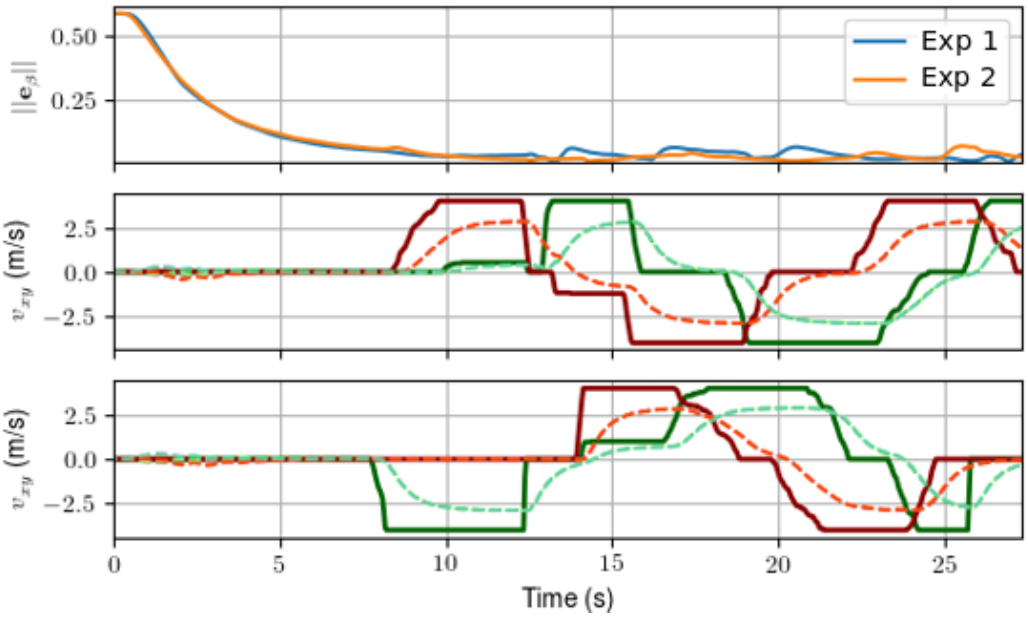


Figure 4.6 – Gazebo simulation of two flights. The solid red and green lines show the desired x and y velocities. The dashed lines show the velocity of each of the three quadrotors which are superposed (all values are expressed in $\mathfrak{F}_0$).

where the centripetal acceleration term $\dot{\psi}_X^2 \mathbf{r}_i$ is needed to counteract the expansion of the formation as it rotates. Care must be taken however if sudden yaw rates such as steps or fast ramps are applied. In such a case the expansion of the formation will only occur once the formation accelerates to the new formation angular velocity set-point $\dot{\psi}_\mathcal{F}$, thus a instantaneous application of the centripetal compensation with respect to the $\dot{\psi}_\mathcal{F}$ will cause the formation to collapse in upon itself, and was the cause of inter-UAV collisions in early simulations. Instead we choose to correct for centripetal acceleration caused by term

$$\dot{\psi}_X = \begin{cases} \omega_{z,i} & \text{if } ||\omega_{z,i}|| < ||\dot{\psi}_\mathcal{F}|| \\ \dot{\psi}_\mathcal{F} & \text{otherwise} \end{cases} \tag{4.29}$$

which calculates the centripetal acceleration as the minimum of the desired formation yaw rate, and UAV yaw rate to be conservative (it is better to expand than to collapse)[2]. Note that we haven't accounted for the acceleration of each robot due to the formation scale $\ddot{s}_\mathcal{F}$ but in our experiments we generally maintain the formation at a constant scale. If high scale-rate dynamics are desired than it could certainly be developed by modifying Eq. (4.28).

2. A video demonstrating the effect of different centrifugal compensation term can be found at: https://box.ec-nantes.fr/index.php/s/WxPKXxeJ3MweLNC

We demonstrate the success of the control and manoeuvring strategy by performing simulated flights (using the gazebo simulator discussed in appendix A) with a three-UAV complete graph formation in two different configurations. The formation altitude, scale, and yaw are regulated by an autopilot, and the x and y components of the trivial motion velocity are set by a joystick. These results shown in Fig. 4.6 show that we are able to manoeuvre the formation quite effectively in the nullspace of the formation control task.

4.2.5 Actuation commands

The SOVS controller generates a control on the second derivative of the flat outputs of the quadrotors, which must be converted to an actuation command. This corresponds to a thrust vector and a yaw torque, however it is difficult to accurately estimate the coefficient of drag and the inertia of the quadrotor, and requires the development of a custom control application on our experimental platform. We therefore integrate the yaw acceleration and pass the resulting thrust vector and yaw rate to the flight controller. Thus the output at time t from the initial control step is transformed as

$$\mathbf{u}(t) = \begin{bmatrix} \mathbf{f}_i^d(t) \\ \int_0^t \ddot{\psi}_i(t)\ \mathrm{dt} \end{bmatrix} \rightarrow \begin{bmatrix} f_i^d \\ \mathfrak{q}_i^d \\ \omega_{z,i}^d \end{bmatrix} \tag{4.30}$$

where the integral of the yaw acceleration is saturated to prevent excessive wind-up. The simple translational dynamic model of the quadrotor presented in section 2.3.1 can be more precisely expressed as

$$\mathbf{f}_i = m_i \left(\mathbf{a}_i - \mathbf{g} \right) + \boldsymbol{\delta}_v \tag{4.31}$$

where $\boldsymbol{\delta}_v$ is an aerodynamic drag force generally proportional the square of the quadrotors airspeed. This can be modelled with some difficulty for each individual UAV [51], [52] but it is easier in our case to simply consider it as a disturbance and modify the lower level quadrotor controller accordingly. From Eqs. 4.24 and 4.28, let us define the desired acceleration of the quadrotor $\mathbf{a}_i^* = \mathbf{a}_i^\beta + \mathbf{a}_i^{\mathfrak{M}}$ as the sum of the bearing control and manoeuvring accelerations. We will then use a PI control with feed forward (FF) to calculate a desired acceleration that should compensate for the unmodelled forces acting on the

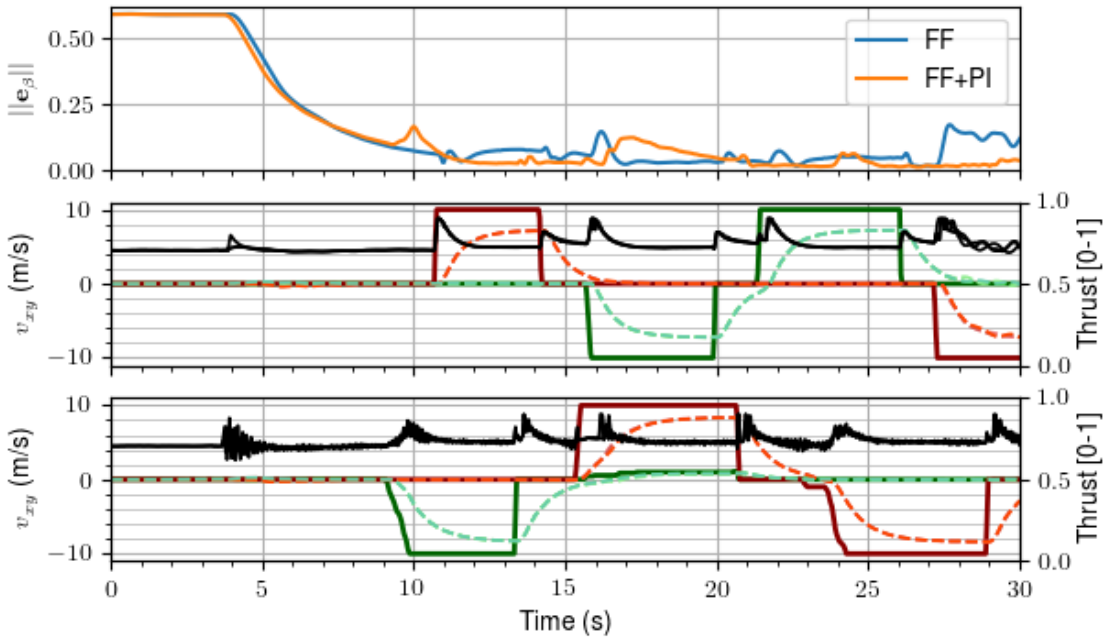


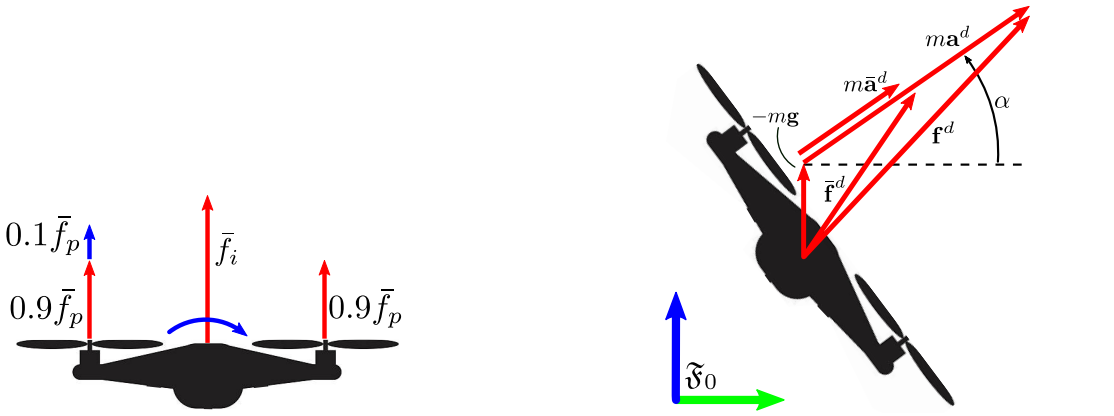
Figure 4.7 – Gazebo simulation comparing two flights with and without a PI controller on the acceleration. The top plot shows the bearing error for the feed-forward (FF) and feed-forward with PI (FF+PI) controllers, and the second and third plots show the desired (solid) and measured (dashed) formation velocities in the x (red) and y (green) directions for FF and FF+PI respectively. The right y-axes show the fraction of maximum thrust (black lines) for each of the UAVs.

quadrotor. The desired acceleration will therefore be

$$\mathbf{a}_i^d = \mathbf{a}_i^* + k_p(\mathbf{a}_i^* - \mathbf{a}_i) + k_i \int_0^t (\mathbf{a}_i^* - \mathbf{a}_i)\mathrm{dt} \tag{4.32}$$

where k_p and k_i are positive gains, which may serve to counteract a certain amount of disturbance and the force vector can be obtained by inserting the resulting desired acceleration into Eq. (4.31), assuming that the integral control eventually compensates for the unknown aerodynamic disturbance.

In Figure 4.7, the effect of using a feedback loop on the acceleration controller is demonstrated in gazebo simulations, by testing two formations with three UAVs and complete bearing measurements graphs. From a low-error initial formation embedding, the bearings are allowed to converge, and then a series of x and y velocity step inputs with magnitudes of 10 ms^{-1} are given to the formation. In the test using simple feed-forward (i.e. $\mathbf{a}_i^d = \mathbf{a}_i^*$), the maximum mean velocity magnitude reached was $v_{x\mathcal{F}} = 7.8$ ms^{-1} in the x-direction and $v_{y\mathcal{F}} = 7.4$ ms^{-1} in the y-direction. With PI feedback (i.e. using the complete Eq. (4.32) with non-zero gains), these values increased to $v_{x\mathcal{F}} = 8.6$ ms^{-1} and



(a) Limiting maximum thrust (red) to maintain a margin for attitude control (blue).

(b) Determining the saturated thrust vector to achieve a linearly scaled acceleration

Figure 4.8 – Thrust saturation handling

$v_{y_{\mathcal{F}}} = 8.4$ ms^{-1} respectively. The improved steady-state error of the FF+PI acceleration controller comes at the expense of a noisier thrust signal, however there was no noticeable chattering or jerky motion in the simulation. We remark however that excessively increasing the gains on the low level controller resulted in an unstable control before the steady state error was eliminated. Therefore if precise velocity control is essential, it may be advisable to estimate and compensate for the model uncertainties along with a positive integral gain.

Along with the low-level acceleration controller, we must also consider the thrust limits of the quadrotor on the output of the SOVS control. We may assume that the quadrotor has some maximum actuation thrust $\bar{f}_i$ which is assumed to be decoupled from the differential thrust required to generate steering moment. This is done by saturating the maximum thrust at an experimentally pre-determined value (in our case 90%) of the sum of the maximum propeller thrusts $\bar{f}_p$. If the formation controller then outputs a thrust such that $f_i^d > \bar{f}_i$ we must saturate the control acceleration to ensure that the quadrotor continues to translate in the desired direction. If one were to simply saturate the thrust along the direction of the thrust vector, the quadrotor would lose both a component of the thrust acting to control its translational acceleration, and a component of the thrust counteracting gravity. This would cause the quadrotor to follow a convex parabolic path instead of a straight line. We therefore need to take some more care in applying the actuation limits of the attitude thrust control.

In the case where $||\mathbf{f}_i^d|| > \bar{f}_i$, the maximum saturated acceleration $\bar{\mathbf{a}}_i^d = \lambda \mathbf{a}_i^d$ s.t. $0 < \lambda < 1$ must be determined such that $||m_i \left(\bar{\mathbf{a}}_i^d - \mathbf{g}\right)|| = \bar{f}_i$. Using the law of cosine, we can find

the equation

$$m_i^2 \left(||\mathbf{g}||^2 + ||\bar{\mathbf{a}}_i^d||^2 - 2||\mathbf{g}||\,||\bar{\mathbf{a}}_i^d|| \cos\left(\frac{\pi}{2} + \alpha\right)\right) = \bar{f}_i^2 \tag{4.33}$$

where α is the elevation angle of the desired acceleration vector, and can be found as

$$\alpha = \cos^{-1}(\mathbf{e}_3^T \mathbf{a}_i^d / ||\mathbf{a}_i^d||) \tag{4.34}$$

which formulates an explicit quadratic formulation of saturated desired acceleration magnitude. Solving for $||\bar{\mathbf{a}}_i^d||$ and then determining the scale factor $\lambda = ||\bar{\mathbf{a}}_i^d||/||\mathbf{a}_i^d||$ allows us to determine the saturated acceleration vector $\bar{\mathbf{a}}_i^d$ and thus the desired force vector which generates the largest achievable acceleration in the desired direction may be obtained from Eq. (4.31). This saturated thrust vector can be remapped into a thrust and attitude as in Eq. (4.30) in order to be sent to the attitude controller.

4.3 Experimental Validation of SOVS

Before testing the formation control algorithms on real UAVs, Gazebo simulations were run with the same firmware to determine tunning parameters and quadrotor trajectories from the approximate initial conditions of the experiments. It was remarked that with large initial bearing errors, the UAVs would tend to move outwards, with the formation expanding before contracting to the desired scale. This expansion of the formation under large errors seems similar in nature to the well-known retraction problem in visual servoing [135] and can be solved by various techniques such as MPC which is applied in the following chapter. Experiments were therefore all started from relatively low bearing errors as the flight volume is very limited. Significant difficulty was encountered in the real-time experiments with the platform described in appendix A and, while some successful flights were achieved, the large majority resulted in crashes, mostly from one or more UAVs crashing into the net during the transient expansion.

Nonetheless, some successful experiments were achieved, the results of which are shown in Fig. 4.9. Both experiments consist of a 3-UAV formation defined by a complete graph, and arranged in the same geometry. An autopilot sets $v_{x\mathcal{F}}$, $v_{z\mathcal{F}}$ and $\dot{s}_{\mathcal{F}}$ in order to keep the formation at a constant scale. In experiment 1 (Exp 1), we use the complete formation control formulation, and observe encouraging initial convergence properties. A steep yaw

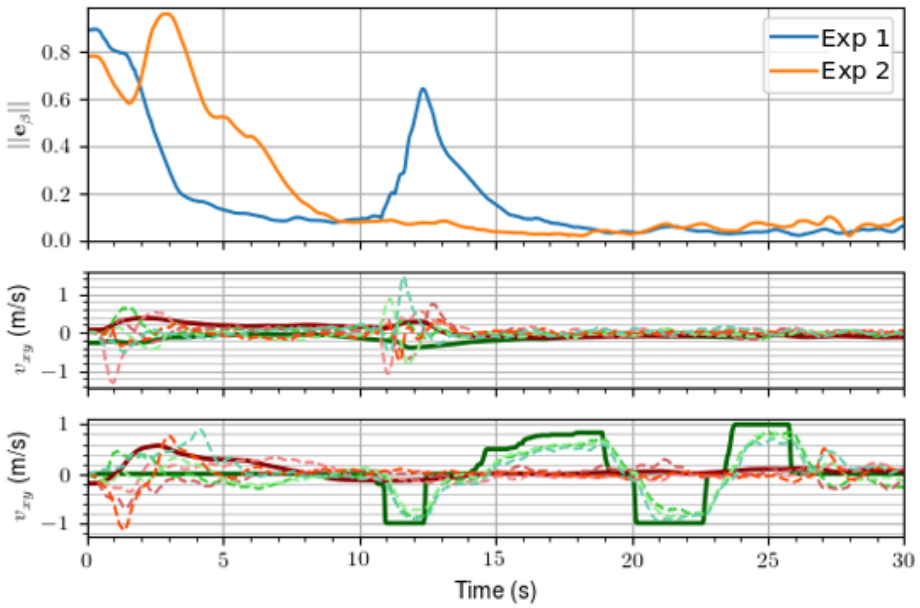


Figure 4.9 – Experimental results for 3-drone SOVS control. The top graph shows the bearing error magnitude per edge for two experiments. The second and third graphs show the x (solid red) and y (solid green) components of the desired formation velocity for experiments exp1 and exp2 respectively. The dashed lines show the x and y velocity components of each UAV.

rate command is applies at 11 s, and it is seen that the formation reacts poorly, nearly becoming unstable. Experiment 2 does not include the hessian vector, and while the initial convergence suffers, it is more stable. The formation is given step velocity commands in the y-direction, and the is able to track these velocities while maintaining a bearing error of less than $0.1/\mathcal{E}$ during the manoeuvres, and less when the desired nullspace velocities are smooth.

All experimental flights using the hessian had noticeable stability issues, while the quadrotors in flights without the hessian appeared much smoother, and had less of a tendency to become unstable when perturbed. This phenomenon did not appear (or was less prominent) in simulations, likely indicating that the hessian is highly sensitive to noise and communication delays, which are much harder to control in real experiments. However despite the difficulties in implementing this controller in real time, it was shown to have good performance with step velocities of 1 ms^{-1} with little coupling with the bearing error.

4.4 Conclusion

SOVS has shown some promising characteristics in manipulator control and furthermore establishes a direct link between bearing measurements and the the quadrotors' actuation thrust vectors. In this chapter we developed the complete non-linear SOVS model on $\mathbb{R}^3 \times \mathbb{S}^1$ accounting for poor numerical conditioning in the interaction matrix as well as actuator limits. Simulations and experiments validated that this controller is feasible, and indeed in simulation it seems to have some interesting properties: it provides good manoeuvring control of the formation while maintaining the desired formation, however the convergence properties of this controller are quite complicated, showing an unfortunate tendency to initially expand during the transient phase.

Real-time experiments showed that this is a valid control method, however the stability of the controller is highly affected by noise and delays. Using the hessian matrix in the control was found to be detrimental to the stability of the controller, likely due its strong non-linearity being greatly affected by noise and delays. While the experimental results remain somewhat unconvincing, better results may have been found if more time had been devoted to the testing and implementation of this controller. Because the controller developed in the following chapter quickly demonstrated better results, more effort was devoted to those experiments and those of this chapter were performed for completeness but less thoroughly. We should note that despite relatively poor convergence properties, in simulation at least it is shown to have certain advantages when controlling formations along high-speed trajectories as is presented later in chapter 6.

CHAPTER 5

MODEL PREDICTIVE BEARING FORMATION CONTROL

ONE of the most promising developments in the control of complex robotics systems in the past decade has been the improvements in model predictive control (MPC). A type of optimal control, it has successfully been applied to humanoid robots, aerial vehicles and other applications. This chapter presents one of the most significant contributions of this thesis, showing how MPC may be applied to bearing formations of quadrotors, to achieve fast-converging formation geometry control along with agressive manoeuvring. The first section details a brief background of MPC, including general theory and its applications

in modern robotics. We then present how the optimization problem may be formulated for bearing formations of quadrotors. Extensive simulations demonstrate that this is in fact a reliable control methodology for bearing formations, and experiments show that this controller allows for fast formation flight. The controller is then extended to show that it is well capable of considering a range of practical operational constraints. The first part of this chapter has been presented at ICRA 2021 [136].

5.1 Background of Model Predictive Control

MPC was developed in the 1970s and onwards, initially for industrial processes with complicated state and input constraints [137], [138]. Instead of directly applying a precomputed relationship between the current and desired states and the control inputs, it models the process over some future time span and then minimizes a cost function generally related to the state error and actuation over the range of the prediction by modifying the control inputs. With the increase in portable computational power, MPC began to be deployed in robotics, and has proven applications in autonomous vehicles [139]–[141], humanoid robots [142]–[145], and aerial robots (more in section 5.1.3). It has furthermore been successfully applied to visual robotics problems, and allows easy integration of mechanical, visual, and task based constraints. In this section, background information is provided on MPC to understand how it works and how it is currently used in aerial robotics and multi-robot systems.

5.1.1 Methodology

We begin by considering a general continuous time system modelled as a function of its state $\mathbf{x}(t)$ and input $\mathbf{u}(t)$ in the typical generic state space form

$$\dot{\mathbf{x}}(t) = f\left(\mathbf{x}(t), \mathbf{u}(t)\right) \tag{5.1}$$

Classical non-linear control consists of finding a set of inputs at any given time which drive the state towards some desired state, such that at any given time one may find a control input that is uniquely determined by the current and desired system states. This controller however does not take into account future factors such as long-term performance, state constraints,

or any behaviour beyond the instantaneous action. If the process model is very complicated and/or if there are constraints on the states or inputs, optimal control methods are often used. Note that optimal control is a vast field, and for the purpose of this thesis, we limit ourselves to non-linear finite-horizon MPC. With a state measurement $\mathbf{x}_0 = \mathbf{x}(t_0)$ at the current time t_0 and by integrating the system model from the current time (without loss of generality, we can let $t_0 = 0$) over a finite prediction horizon T_p, a prediction of the system states at time t may be obtained by

$$\mathbf{x}(t) = \mathbf{x}_0 + \int_0^{T_p} \dot{\mathbf{x}}(t)dt \qquad \text{where } 0 \leq t \leq T_p \tag{5.2}$$

which is a function of the current measured state and the future open-loop control inputs. An objective function $\mathcal{O}(\mathbf{x}(t), \mathbf{u}(t))$ may then be defined to formulate the open-loop control problem as an optimization problem. By solving the constrained optimization problem

$$\min_{\mathbf{u}(t)} \quad \int_0^{T_p} \mathcal{O}(\mathbf{x}(t), \mathbf{u}(t))\, dt \tag{5.3a}$$

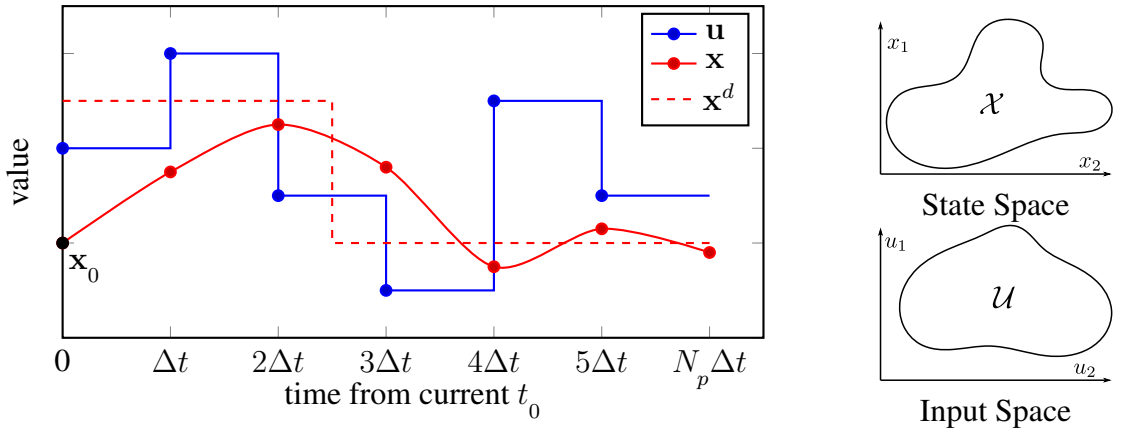
$$\text{s.t.} \quad \dot{\mathbf{x}}(t) = f(\mathbf{x}(t), \mathbf{u}(t)) \tag{5.3b}$$

$$\mathbf{x}(t) \in \mathcal{X} \tag{5.3c}$$

$$\mathbf{u}(t) \in \mathcal{U} \tag{5.3d}$$

$$\forall\, t \in [0, T_p] \tag{5.3e}$$

where $\mathcal{X}$ and $\mathcal{U}$ are the sets of feasible states and inputs respectively (see Fig. 5.1), the sequence of inputs $\mathbf{u}(t)\ \forall t \in [0, T_p]$ that minimizes the integral of the objective function over the whole prediction horizon may be determined. The objective function is therefore chosen so as to minimize the desired state error, and often the control input. It one were to compute the solution to the optimization problem in Eq. (5.3) and provide the resulting control input to the system, a perfectly modelled and measured system will follow the predicted path which minimizes $\mathcal{O}$ over the prediction horizon. We know however that no system is perfectly modelled, as there is always some uncertainty in its initial state, some physical parameters that are neglected or misidentified, and some external perturbations that cannot be predicted. Thus the real path of the system will not be the same as the state prediction subjected to the optimal open-loop control input, and indeed the real state will increasingly deviate from the predicted state as the time from t_0 increases. MPC consists of repeatedly solving the



(a) Optimal input and predicted state evolution for N_p future steps at t_0 (b) Constraint spaces

Figure 5.1 – MPC Control Formulation

open loop optimization given a prediction based on the most recent state measurements and applying the sequence of resulting inputs, until a the solution to a new optimization problem using a new set of measurements as the initial conditions becomes available.

As MPC requires the repeated online solving of optimization problems, it is often limited by computational speed. In fact, an exact minimization of $\mathcal{O}$ with respect to a continuous control input $\mathbf{u}(t)$ is often not possible in a short period of time, and thus by the time the optimal input is calculated, the system would have already diverged from the path of the prediction and the calculated control input would no longer be optimal for the new system state. This also becomes increasingly difficult as the length of the prediction horizon T_p increases. To meet these challenges, a discrete model of the system evolution may be applied, dividing the prediction timespan into N_p discrete time steps of length Δt. The predicted state as a function of a given set of discrete control inputs can be evaluated either analytically or by numerical integration methods. The optimization problem may thus be formulated as

$$\min_{\mathbf{u}_1 \dots \mathbf{u}_{N_p}} \sum_{k=1}^{N_p} \left(\underbrace{\left(\mathbf{x}^d - \mathbf{x}_k\right)^T \mathbf{Q}_k \left(\mathbf{x}^d - \mathbf{x}_k\right)}_{\text{state error cost}} + \underbrace{\mathbf{u}_{k-1}^T \mathbf{R}_{k-1} \mathbf{u}_{k-1}}_{\text{control cost}} \right) + \underbrace{\left(\mathbf{x}_{N_p}^d - \mathbf{x}_{N_p}\right)^T \mathbf{Q}_{N_p} \left(\mathbf{x}_{N_p}^d - \mathbf{x}_{N_p}\right)}_{\text{terminal error cost}} \tag{5.4}$$

where $\mathbf{Q}_k$, and $\mathbf{R}_k$ are positive diagonal matrices containing gains (or stage costs) on the process over time, and $\mathbf{x}_k = \mathbf{x}(t_0 + k\Delta t)$. The terminal error $\mathbf{x}_N^d - \mathbf{x}_N$ evaluated at the prediction horizon is often given a larger gain (or terminal cost) by the positive diagonal matrix $\mathbf{Q}_{N_p}$ to encourage a complete convergence of the error by the end of the prediction horizon, usually at the expense of a greater control effort.

5.1.2 Optimization Algorithms

MPC relies on the repeated solving of an open-loop optimal control problem, and this may be decomposed into two primary steps

1. Predicting the state evolution with respect to the initial state and a sequence of future control variables

2. Optimizing the control variables to minimize the predicted objective function and respecting the necessary constraints

The first step consists of finding the solution to the initial value problem (IVP) over a finite prediction horizon $\mathbf{x}(t)\ \forall t \in [0, T_p]$, where $\mathbf{x}(t)$ is uniquely defined by the system model in Eq. (5.1) and current state measurement or estimate $\mathbf{x}_0$ [146]. The simplest way of solving an IVP problem is using the so-called single-shooting method, for which the solution can be determined by a numerical integration of Eq. (5.2). This method however is subject to long computation times as the operations must be performed in series, and is known to have poor stability as it is highly affected by noise on the initial condition and by the discretization of the time [147]. Multiple shooting has become a more popular method of solving the IVP and has many benefits compared to single shooting. It consists of decomposing the original IVP into N intervals and solves the boundary value problem (BVP) $\mathbf{x}_n(t) \in [t_{n-1}, t_n]$ for each subinterval n. The solved sub-intervals are then fitted together by introducing constraints on continuity forming a more numerically stable solution to the IVP. Beyond numerical stability, a major benefit of multiple shooting is the fact that the BVP of each subinterval may be computed in parallel, greatly reducing the final computation time. While this comes at the expense of a much more complicated implementation, numerous libraries have been developed for implementing multiple shooting and thus details on the implementation are not required. There are also newer state of the art integration methods such as collocation [148] which are not used in this thesis, but have been successfully used in other MPC implementations. For the MPC algorithms presented in this chapter, the matlab/simulink simulations make use of single-shooting with 4th order Runge-Kutta (RK4) integration, while Gazebo simulations and experiments make use of multiple shooting, which is the current state-of-the-art IVP solver for MPC problems.

Having solved the IVP of the system model, the next main step consists of optimizing the control variables such that the objective function is minimized. As with the prediction

component of MPC, non-linear constrained optimization is a well-studied field with applications in many fields of mathematics and engineering, we therefore only give a brief overview here. Because MPC requires rapid and guaranteed convergence, typically Newton methods are used to minimize the objective function, although there are some works such as [149] which consider global optimization approached. Of these, the two most popular are sequential quadratic programming (SQP) and interior point methods, the former of which is briefly outlined: Given a function $f(\mathbf{x})$ subject to constraint vectors $\mathbf{g}(\mathbf{x}) \leq \mathbf{0}$ and $\mathbf{h}(\mathbf{x}) = \mathbf{0}$, the optimization which is typically formulated as

$$\min_{\mathbf{x}} \quad f(\mathbf{x}) \tag{5.5a}$$

$$\text{subject to} \quad \mathbf{g}(\mathbf{x}) \geq \mathbf{0} \tag{5.5b}$$

$$\mathbf{h}(\mathbf{x}) = \mathbf{0} \tag{5.5c}$$

may be reformulated as a Lagrangian function combining all objectives and constraints

$$\mathcal{L} = f(\mathbf{x}) + \boldsymbol{\mu}^T \mathbf{g}(\mathbf{x}) + \boldsymbol{\lambda}^T \mathbf{h}(\mathbf{x}) \tag{5.6}$$

where $\boldsymbol{\lambda}$ and $\boldsymbol{\mu}$ are vectors of Lagrange multipliers. This transforms the optimization problem to a simple quadratic programming problem with linearized constraints

$$\min_{\mathbf{d}} \quad \boldsymbol{\nabla} f(\mathbf{x})^T \mathbf{d} + \frac{1}{2} \mathbf{d}^T \boldsymbol{\nabla}^2 \mathcal{L} \mathbf{d} \tag{5.7a}$$

$$\text{subject to} \quad \mathbf{g}(\mathbf{x}) + \boldsymbol{\nabla} \mathbf{g}(\mathbf{x})^T \mathbf{d} \geq \mathbf{0} \tag{5.7b}$$

$$\mathbf{h}(\mathbf{x}) + \boldsymbol{\nabla} \mathbf{h}(\mathbf{x})^T \mathbf{d} = \mathbf{0} \tag{5.7c}$$

The new variable can then be chosen as a distance along the calculated direction

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{d} \tag{5.8}$$

where the step length $\alpha_k \in]0, 1]$ can be chosen according to many different methods, with $\alpha_k = 1$ corresponding to a full Newton step [150]. The SQP optimization algorithm therefore transforms the optimization problem into a set of quadratic programming sub-problems which are sequentially solved. The other popular method for non-linear optimization is the "Interior Point" method, which differs from SQP algorithm by including the inequalities as

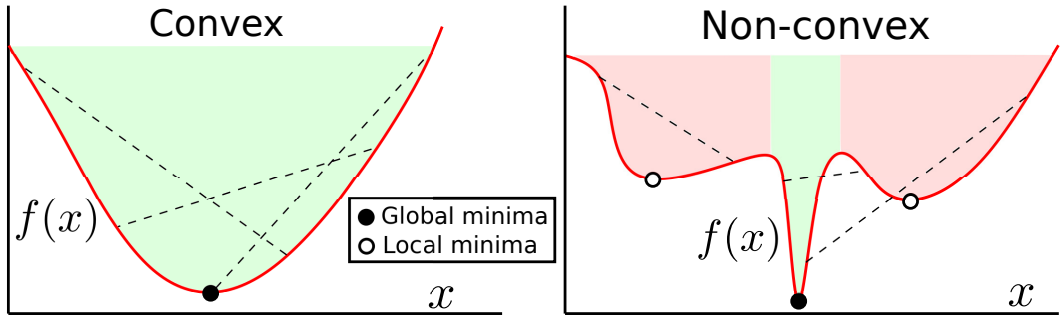


Figure 5.2 – Qualitative examples of convex and non-convex functions. Note that if the initial condition for the optimization lies in the red area, the solution will tend to converge to a local minima, while initialization in the green area will tend to converge at the global minima.

penalties on the minimization of $f(\mathbf{x})$. While it is used in some MPC software, most robotics applications tend to favour SQP as discussed hereafter. Note that both these algorithms are gradient-based methods which inherently rely on a problem being convex (i.e. a line between any two points on the function will never cross the function boundaries, see Fig. 5.2). If the function and constraints are not convex, convergence to the global minima cannot be guaranteed.

Because optimization is often a time-consuming process, adaptations specific to MPC have been developed to improve real-time performance. The real time iteration (RTI) method takes advantage of the structure of the SQP algorithm to separate the control problem into two steps, 1) a preparation step and 2) a feedback step. Taking advantage of the fact that the previous SQP solution $(\mathbf{x}_k, \mathbf{u}_k)$ will be close to the solution at the current state, the preparation phase calculates the gradients, hessians, and sensitivity matrices, and once a new state measurement is available the feedback step will only need to update the initial condition of the IVP, and the quadratic program may be solved very quickly [151]. If the predicted state is accurate, the initial guess of the optimization variables will align with the optimal solution, however this will never be the case due to measurement error, model perturbations, and discretization. Because of this, there will be some error in the preparation step, and the reduced sensitivity to this error is one of the reasons for preferring SQP algorithms over interior point [152].

5.1.3 Model Predictive Control For UAVs and MRSs

Because of its practicality in integrating constraints into control laws, MPC began to be applied to multi-rotor UAVs early in the 2010's once onboard computing became powerful

enough to handle the calculations. While we give a brief overview of MPC in aerial robotics, the reader is referred to a recent survey on the matter for more details [153]. Works such as [154]–[157] used linear MPC with the model linearized around the hovering state to stabilize the outer loop in a hierarchical quadrotor controller (e.g. for tracking trajectories, flying in formation), using classical techniques such as PID and feedback linearization for the attitude control. Later works such as [158], [159] moved towards non-linear MPC as computational hardware and algorithms evolved. A detailed comparison [160] showed that while the computation times required for non-linear MPC were almost an order of magnitude slower than its linear counterpart, it was able to achieve significantly better results when following aggressive trajectories or in the presence of disturbances, when using the attitude and thrust as control inputs. Non-linear MPC in particular has also shown the ability to perform online planning to fly UAVs through statically unstable configurations, bypassing obstacles (narrow non-horizontal channels for example) that would not be possible with purely reactive controllers [159], [161].

In the past couple years, MPC (particularly non-linear) has become quite prevalent in the field of aerial robotics. In fields such as aerial manipulation, MPC is used to provide accurate force control and physical interaction constraints with the environment [75], [162], [163]. While most of the early work on MPC dealt only with the translational dynamics of quadrotors, many of the recent aerial manipulation controllers operate on the actuator level (i.e. motor speeds) of omnidirectional UAVs removing the need for a cascading controller which allows a more complete use of the UAV's dynamic capabilities. Indeed in some cases [35], [164] the system model goes even further, using the time derivative of the thrust of each motor as the optimization variable. This allows constraints such as the rise time of the propeller (limited by aerodynamic drag and motor torque) to be accounted for in the controller. Because of the complexity of the model, the fast solution times required, and the high required control frequency to reduce vibrations when controlling the propeller speed of UAVs, these are run on a powerful offboard computers with wired data transmission.

Beyond its use for trajectory tracking and aerial manipulation, MPC has been identified as an attractive solution for visual control problems with robotic arms due to the ability to account for actuator, sensing, and task constraints [135], [165], [166]. This was also adapted to quadrotors, for which the predictive nature of MPC is particularly important, as the underactuated nature tends to increase the difficulty of visual servoing control by classical

methods. In [167], a visual tracking solution for a quadrotor-mounted monocular camera uses B-spline relative coordinates to generate a time-optimal trajectory following a moving target, while avoiding occlusion and collisions. Another work [168] considered visual tracking as task decoupled from the trajectory tracking task. MPC was used to calculate the body-frame angular velocities and thrust for a quadrotor, tracking an externally specified trajectory while pointing the camera at a given spatial point. This concept was then extended by [169], which combined the work of [168] and [164] to use MPC on the propeller thrust derivatives of an omnidirectional multirotor to track multiple visual features. Other works treating MPC for UAV vision-based tasks include [170] for which a quadrotor is controlled by an MPC algorithm optimizing the vertical velocity, roll, pitch, and yaw rate, while tracking features that statistically will give the best localization results. Finally, a detailed analysis of the stability of image-based visual serving by a quadrotor using MPC is provided in [171]

Moving from UAV control to multi-robot system control, MPC has been applied to solve constrained swarming and formation control tasks. As early as 2004, MPC was used for trajectory tracking and mutual collision avoidance between multiple UAVs. This was tested with both centralized MPC (a single controller with access to all states $\mathbf{x}_1 \dots \mathbf{x}_n$ and optimizes all inputs $\mathbf{u}_1 \dots \mathbf{u}_n$) and decentralized (or distributed) MPC [172]. The latter consisted of a controller for each robot with the i^{th} controller having knowledge of state $\mathbf{x}_i$ and of a subset of neighbour states $\mathbf{x}_j \ \forall j \in \mathbb{N}_i$, and which optimized only the input $\mathbf{u}_i$ of $\mathcal{A}_i$. The computation time for the centralized controller was vastly larger than that of the decentralized controller with little difference in operational performance. In some cases such as with small formation, the centralized control runtime isn't an issue [173], however it clearly doesn't scale well and requires continuous uninterrupted bi-directional communication with a centralized controller when local measurements are used. When decentralized optimization routines are carried out to completion for each subsystem with neighbours exchanging state predictions periodically, the distributed solution will be equal to the exact solution [174]. This is may be possible for slow systems with high computational power and reliable networking (such as chemical processes) but is infeasible in dynamic robotic systems and is not coherent with the RTI method. Instead what is more practical for multi-robot systems is a distributed MPC where each robot uses an estimate of the evolution of its neighbours at each RTI step, possibly communicating its state or optimization results to its neighbours to help with their next RTI step. This strategy is applied to simulations

of position-based quadrotor formation flight in [175], [176] with external collisions and inter-robot distance constraints, where the MPC controllers plan and share spline trajectories that their low level controllers may then track. MPC is also used in real experiments for UAV swarms in [177] where a centralized MPC controller navigates the UAVs through a cluttered environment avoiding obstacles and inter-UAV collisions while maintaining swarm coherence. Likewise in [178], decentralized MPC provides planning for formations of up to 30 UAVs by calculating trajectory points in a bottlenecked environment. To the best of our knowledge however, MPC has never been implemented in a decentralized manner and applied to the bearing formation control problem on the $\mathbb{R}^3 \times \mathbb{S}^1$ manifold, generally using relative positioning when applied to multi-robot systems.

5.2 Model Predictive Formation Bearing Control

With the success of previous model predictive controllers applied to UAVs, we believed that it would be of interest to apply this type of controller to bearing formation control. This would allow a direct link between the lower level and highly non-linear attitude dynamics of drone and the evolution of the bearing features. This would allow the use of a more accurate model of the drone in the controller, allowing the gains to be increased without compromising stability. Furthermore, the implicit optimality of MPC may lead to a better formation control response, with faster bearing error convergence and high-speed manoeuvring. In addition to providing potentially faster control by considering a more accurate robot model, MPC also deals well with constraints, and thus the incorporation of tasks of practical importance outside of the typical formation control problem such as obstacle avoidance, sensor limits, and potentially even rigidity maintenance can be relatively easily incorporated once the initial objective formulation is developed.

5.2.1 Formulation of the Optimization Problem

The first step in the design of a model predictive controller is the formulation of the optimization problem, including the choice of control variables, objectives, and constraints. In the formulation of the problem, we must consider the complexity of the optimization (for a reasonable execution time) as well as the accuracy of the plant model so that the optimized control variables will have a similar effect on the output as predicted.

5.2.1.1 Control variables

The choice of control variable is an important decision, which depends partially on the optimization problem, but also on the means of implementation. The hierarchical structure of multirotor controllers means that there are many available options. We recall here that the dynamic model of a multirotor is

$$\dot{\mathbf{p}}_i = \mathbf{v}_i \tag{5.9a}$$

$$\dot{\mathbf{v}}_i = \frac{1}{m_i} f_i \begin{bmatrix} 2(q_w q_y + q_x q_z) \\ 2(q_y q_z - q_w q_x) \\ 1 - 2(q_x^2 + q_y^2) \end{bmatrix} + \mathbf{g} \tag{5.9b}$$

$$\dot{\mathfrak{q}}_i = \frac{1}{2} \begin{bmatrix} 0 \\ \boldsymbol{\omega}_i \end{bmatrix} \otimes \mathfrak{q}_i \tag{5.9c}$$

$$\dot{\boldsymbol{\omega}} = \mathbf{I}^{-1}(\boldsymbol{\tau}_i - \boldsymbol{\omega} \times \mathbf{I}\boldsymbol{\omega}) \tag{5.9d}$$

Note that this formulation expresses the attitude as a quaternion instead of a rotation matrix, which is chosen as the attitude representation in this chapter due to the lighter computation cost of quaternion operations. We furthermore note that this dynamic model does not actually represent the true actuation means of the multi-rotor which is in fact the propellers velocities $\boldsymbol{\Omega}$. With the presented model, the lowest-level actuation available is the actuation wrench $[f_i \ \boldsymbol{\tau}_i]$ which is a linear mapping of the squared propeller speeds. However the propellers cannot instantaneously change speeds, and are often modelled as a first order system, thus the derivative of the propeller speeds could be considered as inputs, at was done in [35]. We could continue to go deeper into the electrical and aerodynamics of the multi-rotor actuation however the dynamics of this would be impractically fast for a small multirotor, making it difficult to solve the optimization at sufficient speed.

Selecting the appropriate control variable depends on the expected dynamics of the model. Choosing a high-level control such as the velocity of the quadrotor or spline trajectory coordinates will mean that the prediction of the state evolution will be accurate, and the optimization will be relatively quick and simple. It suffers however from being unaware of all lower-level closed-loop responses and possible constraints such as the propeller speed limits and attitude control response times. The prediction will therefore not be an accurate representation of the open-loop dynamics of the controller, as the velocity-tracking control of

the quadrotor will not necessarily be able to enforce the optimal velocity trajectory calculated by the optimization problem. This issue is solved by using lower-level control inputs, such as the quadrotor's propeller speeds. In this case, we know that the time constant for the propeller speeds tracking a desired reference is much faster than that of the velocity (although the propeller dynamics slow down as they become larger, for small quadrotors they may be neglected). If we can assume that the propellers are able to quickly track some reference speed, then the only actuation constraints on the system are the maximum and minimum propeller speeds and it is possible to achieve a truly optimal open-loop control. Using the lower level control inputs as optimization variables also have their drawbacks however, as the prediction of the output requires more integration, and thus more accumulation of error due to sensor noise and modelling errors. Furthermore, as the lower-level closed loop dynamics are faster, the optimization needs to use smaller time steps, resulting in either an optimization problem of increased dimension (which must be solved at a higher frequency for closed loop stability) or in a shorter prediction horizon T_p.

The best choice of control variable is therefore the lowest-level control variable for which the higher-level model parameters may be be estimated with a reasonable degree of certainty, and the optimization may be solved sufficiently fast to ensure close-loop stability. In [35] where the propeller angular acceleration is used as the optimization value (for a large hexarotor), the MPC loop runs at 200 Hz on an onboard computer with wired-connections. Most quadrotors in fact have their inner control loops (motor and attitude) running at at least 200 Hz to ensure stable, reactive, and vibration-free flight. These requirements seem to eliminate the technological feasibility of propeller acceleration or speed (or actuation wrench as it is a linear mapping of the propeller speed) as the control inputs for our platform, leaving us to choose some possible actuation inputs in equations 5.9(a-c). What we ended up choosing was the thrust f_i and the desired body-frame angular velocity $\boldsymbol{\omega}_i$ for actuating the quadrotor as is used in [168], and which ran successfully at 100 Hz. As we are only considering small quadrotors, the thrust is assumed to be provided very quickly, while the angular velocity has a rise time of around 50-75 ms (see Fig. 2.9).

5.2.1.2 State prediction

Having chosen the control variables, they must then be related to the outputs of interest. Given the control input $\mathbf{u}_i = [f_i, \boldsymbol{\omega}_i]$, the predicted state of $\mathcal{A}_i$ can be expressed as

$$\mathfrak{q}_i(t) = \mathfrak{q}_i(0) + \int_0^t \dot{\mathfrak{q}}_i dt \tag{5.10a}$$

$$\mathbf{v}_i(t) = \mathbf{v}_i(0) + \int_0^t \dot{\mathbf{v}}_i dt \tag{5.10b}$$

$$\mathbf{p}_i(t) = \mathbf{p}_i(0) + \int_0^t \dot{\mathbf{p}}_i dt \tag{5.10c}$$

for $t \in [0, T_p]$ and can be calculated analytically or using numerical methods. We note that the initial quaternion $\mathfrak{q}_i(0)$ is evaluated from the known roll and pitch of $\mathcal{A}_i$, and its yaw estimated relative to the common reference frame available through a decentralized bearing consensus algorithm. The initial velocity $\mathbf{v}_i(0)$ is equally measured in $\mathfrak{F}_i$ using optical flow and inertial sensor fusion, and projected on the $\mathbb{R}^3 \times \mathbb{S}^1$ manifold. We notice however that there is no available absolute measurement, or even estimation, of $\mathbf{p}_i(0)$ in a frame common to all agents. It must be recalled however that we are not trying to regulate the inter-agent positions, but simply the inter-agent bearings, and the steering component of the controller regulates the distance between the agents. We must therefore find the prediction $\boldsymbol{\beta}_i(t) = [\boldsymbol{\beta}_{i1}(t) \dots \boldsymbol{\beta}_{in}(t)]$ which may be computed in one of two methods:

$$\boldsymbol{\beta}_{ij}(t) = \mathbf{R}_i^T(\psi_i(t)) \frac{\mathbf{p}_j(t) - \mathbf{p}_i(t)}{\hat{d}_{ij}(t)} \tag{5.11a}$$

$$\boldsymbol{\beta}_{ij}(t) = \boldsymbol{\beta}_{ij}(0) + \int_0^t \dot{\boldsymbol{\beta}}_{ij} dt \tag{5.11b}$$

where $\dot{\boldsymbol{\beta}}_{ij}$ may be estimated using the bearing rigidity matrix (which is dependant on the inter-agent distance estimate $\hat{d}_{ij}$). Both options are tested, and it was found that Eq. (5.11a) performed better, and was chosen as the applied method. Predicting the bearings using Eq. (5.11b) was found to work well for some configurations and not on others, likely indicating that it is less numerically stable.

By assuming at every prediction step that the origin of the predicted position is the flat frame aligned with $\mathfrak{F}_i$, we can estimate the initial position of $\mathcal{A}_j$ as

$$\mathbf{p}_j(0) = \boldsymbol{\beta}_{ij}(0)\hat{d}_{ij} \tag{5.12}$$

The position and yaw of $\mathbf{p}_i(t)$ is determined relative to its current position as a function of the optimization variables, and $\mathbf{p}_j(t)$ can reasonably be estimated on of two ways:

- $\mathcal{A}_j$ could communicate its predicted path from its previous RTI prediction step to $\mathcal{A}_i$
- $\mathcal{A}_i$ could assume that $\mathcal{A}_j$ is moving as if it were reacting to the desired steering command $\mathfrak{M}^d = [\mathbf{v}_{\mathcal{F}}^d \ \dot{\psi}_{\mathcal{F}}^d \ \dot{s}_{\mathcal{F}}^d]$.

The first option requires a continuous and time-sensitive exchange of fairly large packets of information, whereas the second requires no addition inter-agent communication beyond what is necessary for the inter-agent yaw and centroid consensus estimations (see section 3.4.4). We therefore take the second assumption and assume that the observed quadrotors' motions over the prediction time is purely based on the velocity resulting from the desired trivial motion. Neglecting modelling errors and noise which makes it impossible to exactly follow a trajectory, an estimation of the motion of $\mathcal{A}_j$ will have two major sources of inaccuracy: the transient motion of $\mathcal{A}_j$ correcting for its own bearing measurements is not accounted for, and the desired steering control will likely not remain constant over the control horizon. The first can only be corrected for by sharing predicted paths between the UAVs at each iteraction of the controller, and the second is only accurate if the sequence of trivial motion commands is known over the full prediction horizon, which is possible for pre-planned formation (infrastructure inspection tasks for example) but not for reactive tasks such as pursuits. In the later scenario, if we imagine the operation of a formation, it is likely that the different components of $\mathfrak{M}^d$ will have different purposes. The linear velocity $\mathbf{v}_{\mathcal{F}}^d$ is used for moving the formation through space, and may be constant for large periods of time. The yaw and scale rates ($\dot{\psi}_{\mathcal{F}}^d$ and $\dot{s}_{\mathcal{F}}^d$) of the formation however are likely used to marginally correct the heading and scale of the formation to a desired value. Large values of $\dot{\psi}_{\mathcal{F}}^d$ and $\dot{s}_{\mathcal{F}}^d$ at $t = 0$ will not necessarily imply in large values at the end of the prediction horizon, thus we use a damped estimate for the predicted future steering commands

$$\begin{aligned} \mathbf{v}_{\mathcal{F}}(t) &= \mathbf{v}_{\mathcal{F}}(0) \\ \dot{\psi}_{\mathcal{F}}(t) &= \dot{\psi}_{\mathcal{F}}(0) e^{\frac{-t}{\tau}} \\ \dot{s}_{\mathcal{F}}(t) &= \dot{s}_{\mathcal{F}}(0) e^{\frac{-t}{\tau}} \end{aligned} \tag{5.13}$$

where the time constant $\tau > 0$ is tuned to be large if the formation is likely manoeuvred along large orientations and changes in scale, and tuned to be small if the formation is likely

to be held at a constant (unpredictably changing) scale and heading. The predicted position of $\mathcal{A}_j$ relative to the initial position of $\mathcal{A}_i$ can therefore be estimated by $\mathcal{A}_i$ to be

$$\mathbf{p}_j(t) = \boldsymbol{\beta}_{ij}\hat{d}_{ij} + \int_0^{T_p} \mathbf{R}_i^T \left(\mathbf{v}_{\mathcal{F}}^d(t) + \dot{\psi}_{\mathcal{F}}^d [\mathbf{z}_0]_\times \left(\mathbf{c}_{\mathcal{F}} - \mathbf{p}_j \right) + \dot{s}_{\mathcal{F}}^d \left(\mathbf{c}_{\mathcal{F}} - \mathbf{p}_j \right) \right) dt \tag{5.14}$$

where the formation centroid can be estimated at $t = 0$ and evolves as $\dot{\mathbf{c}}_{\mathcal{F}}(t) = \mathbf{R}_z(\psi_{\mathcal{F}}(t))\mathbf{v}^d(t)$ with respect to the formation centroid at $t = 0$. Given an initial bearing measurement, state estimate, and future control input, $\mathcal{A}_i$ is thus able to estimate $\mathbf{p}_i(t)$, $\mathbf{p}_j(t)$, and $\psi_i(t)$ for all $0 \leq t \leq T_p$ relative to $\mathfrak{F}_i$ projected onto the $\mathbb{R}^3 \times \mathbb{S}^1$ manifold at the time of the prediction. Given that the yaw can be related to the quaternion attitude as $\psi = \text{atan2}(2q_w q_z + 2q_x q_y, 1 - 2q_y^2 - 2q_z^2)$, the projection of the bearing on the $\mathbb{R}^3 \times \mathbb{S}^1$ manifold at any time within the prediction horizon may therefore be expressed as

$$\boldsymbol{\beta}_{ij}(t) = \begin{bmatrix} \dfrac{(p_{y,j} - p_{y,i})\ (2\,q_w\,q_z + 2\,q_x\,q_y) - (p_{x,j} - p_{x,i})\ (2\,{q_y}^2 + 2\,{q_z}^2 - 1)}{d_{ij}\sqrt{(2\,{q_y}^2 + 2\,{q_z}^2 - 1)^2 + (2\,q_w\,q_z + 2\,q_x\,q_y)^2}} \\ -\dfrac{(p_{x,j} - p_{x,i})\ (2\,q_w\,q_z + 2\,q_x\,q_y) + (p_{y,j} - p_{y,i})\ (2\,{q_y}^2 + 2\,{q_z}^2 - 1)}{d_{ij}\sqrt{(2\,{q_y}^2 + 2\,{q_z}^2 - 1)^2 + (2\,q_w\,q_z + 2\,q_x\,q_y)^2}} \\ \dfrac{p_{z,j} - p_{z,i}}{d_{ij}} \end{bmatrix} \tag{5.15}$$

where $\mathbf{p}_i(t) = [p_{x,i}\ p_{y,i}\ p_{z,i}]^T$, $\mathbf{p}_j(t) = [p_{x,j}\ p_{y,j}\ p_{z,j}]^T$ and $\mathfrak{q}_i(t) = [q_w\ q_x\ q_y\ q_z]^T$. As each robot is able to predict the evolution of the formation, there is now sufficient information to design the formation controller objectives.

5.2.1.3 Optimization objectives

Formation control has two primary objectives: tracking the specified inter-agent geometry defined by the desired bearing vector $\boldsymbol{\beta}^d$, and manoeuvring the formation through its environment by its trivial motions $\mathfrak{M}(t) \in \ker(\mathbf{B})$. In first-order formation control (see chapter 3), it is proven that for a first-order system, the second task lies in the nullspace of the first task. The two tasks are therefore decoupled and may be performed independently. For higher-order models this is not the case. Considering the input $\mathbf{u}_i = [f_i\ \boldsymbol{\omega}_i]$, if we were to solve one optimization to quickly reach a set of desired bearings $\boldsymbol{\beta}^d$ and another to quickly reach the desired trivial motions $\mathfrak{M}^d$, the sum of those control inputs solutions would likely result in a set of inputs adapted to neither one nor the other of the two tasks. We therefore must consider this as a multi-objective optimal control problem, where the two tasks are optimized concurrently with a common set of control variables. We will therefore choose to

formulate the optimization cost function as

$$\mathcal{O} = \mathcal{O}_{\beta} + \mathcal{O}_{\mathfrak{M}} + \mathcal{O}_{\mathbf{u}} \tag{5.16}$$

where the cost function $\mathcal{O}_{\beta}$ penalizes the bearing error, $\mathcal{O}_{\mathfrak{M}}$ penalizes the trivial motion error, and $\mathcal{O}_{\mathbf{u}}$ penalizes excessive actuations, acting effectively as a damping criteria.

We start be defining the steering component $\mathcal{O}_{\mathfrak{M}}$ of the objective function as elements of it will be used when later defining the bearing cost function. We recall that there is a mapping $\mathfrak{M} \mapsto [\mathbf{v}_{i}\ \omega_{z,i}] \forall i$ from the steering command to the flat output of each agent defined by

$$\mathbf{v}_{i}^{d} = \mathbf{R}_{i}^{T}(\mathbf{v}_{\mathcal{F}} + \dot{\psi}_{\mathcal{F}}(\mathbf{c}_{\mathcal{F}} - \mathbf{p}_{i}) \times \mathbf{z}_{0} + \dot{s}_{\mathcal{F}}(\mathbf{c}_{\mathcal{F}} - \mathbf{p}_{i})) \tag{5.17a}$$

$$\omega_{z,i}^{d} = \dot{\psi}_{\mathcal{F}} \tag{5.17b}$$

and that the rotation $\mathbf{R}_{i}$ and the formation centroid $\mathbf{c}_{\mathcal{F}}$ is known with respect to a common frame estimated through a consensus algorithm. Thus any steering motion given to the formation may be decomposed into a flat output for each agent, as is done with classical formation control. The trivial motion tracking error at time t along the prediction is thus

$$\mathbf{e}_{\mathfrak{M}i}(t) = [\mathbf{v}_{i}^{d}(t)^{T}\omega_{z,i}^{d}(t)]^{T} - [\mathbf{v}_{i}(t)^{T}\omega_{z,i}(t)]^{T} \tag{5.18}$$

where $v_{i}(t)$ and $\omega_{z,i}(t)$ are obtained by solving the IVP. A least-square objective function

$$\mathcal{O}_{\mathfrak{M}i} = \sum_{k=1}^{N_p} \mathbf{e}_{\mathfrak{M}i}^{T}(k\Delta t)\mathbf{Q}_{\mathfrak{M}i}\mathbf{e}_{\mathfrak{M}i}(k\Delta t) \tag{5.19}$$

where $\mathbf{Q}_{\mathfrak{M}i}$ is a positive diagonal gain matrix, can then be used to represent the cost of each robot $\mathcal{A}_{i}$ deviating from the desired trivial motion command.

The bearing error component of the objective function can be formulated using the standard quadratic formulation presented earlier, however it is first necessary to predict the bearing evolution for a given set of inputs. We have shown previously that for a known control signal and initial state, we are able to recover a prediction of the state (that is to say the attitude, velocity, and position) of the quadrotor over the prediction horizon. Staying true to the hypothesis that only information from onboard sensors may be used, it is reasonable to assume that the initial conditions of the velocity and attitude ($\mathbf{v}_{0}$ and $\mathfrak{q}_{0}$ respectively)

are reasonably correct, while the evolution of each bearing $\boldsymbol{\beta}_{ij}$ can be obtained from the combination of Eqs. 5.15, 5.14, and 5.10a-c. A least-squares bearing objective function is then defined as

$$\mathcal{O}_{\boldsymbol{\beta}i} = \sum_{k=1}^{N_p} \mathbf{e}_{\boldsymbol{\beta}i}^T(k\Delta t)\mathbf{Q}_{\boldsymbol{\beta}i}\mathbf{e}_{\boldsymbol{\beta}i}(k\Delta t) \tag{5.20}$$

where $\mathbf{Q}_{\boldsymbol{\beta}i}$ is a positive diagonal matrix. Because the manoeuvring and bearing control tasks are not independent on this level of actuation, the relative values between the gains in the $\mathbf{Q}_{\mathfrak{M}i}$ and $\mathbf{Q}_{\boldsymbol{\beta}i}$ matrices will define the point along the pareto-frontier of the two tasks, prioritizing the higher weight. If accurate bearing control is essential, that its gains should be larger, but with a resulting negative impact on the manoeuvring.

The last control objective is on the control input to minimize the action of the controller. We set the "desired actuation" to be the least effort actuation at the hovering state of the quadrotor

$$\mathbf{e}_{\mathbf{u}i} = [m_i g\ \mathbf{0}_3]^T - \mathbf{u}_i \tag{5.21}$$

The optimization should therefore try to minimize the magnitude of the angular velocity and thrust of the quadrotor using the objective

$$\mathcal{O}_{\mathbf{u}i} = \sum_{k=1}^{N_p} \mathbf{e}_{\mathbf{u}i}^T(k\Delta t)\mathbf{Q}_{\mathbf{u}i}\mathbf{e}_{\mathbf{u}i}(k\Delta t) \tag{5.22}$$

This is contradictory to the previous objectives, as if the quadrotor only hovers, it will be unable to reduce the bearing and velocity errors. This objective is then given a low gain with respect to the others, so that the formation will behave as aggressively as possible, and will only have a significant impact when the formation is hovering.

5.2.1.4 Input Constraints

Constraints are used to limit the solution set of the optimization problem to that of a physically feasible system. These can be divided into two categories: input constraints and path constraints. Because input constraints are simple and necessary for all implementations of MPC, we discuss them here, and path constraints are included later on in section 5.6 for situations where they are required. The input constraints in this case are limits on the actuation of the quadrotor that must be respected by the optimization.

While there is a coupling between the available steering torque and maximum thrust,

this occurs at the motor speed level. Finding the absolute physical actuation limits of the quadrotor would therefore require very accurate estimation of the moment of inertia and the propeller thrust and drag coefficients, therefore we choose to simply use conservatively chosen box constraints with empirically chosen limits. Given that the control inputs are thrust and body-frame angular velocity, we simply saturate the maximum thrust at some value $\overline{f}_i$ less than the sum of the full thrust of the motors, leaving a margin of thrust left over to perform the differential body-rate control. The body-rate limits are divided into rangular velocities around the $\mathbf{x}_i$ and $\mathbf{y}_i$ axes ω_{xy} (which will be the same so long as the quadrotor is symmetric) and the yaw rate ω_z. These are determined by performing aggressive manual flights with the maximum thrust saturated at $\overline{f}_i$ and incrementally increasing the saturation levels $\overline{\omega}_{xy}, \overline{\omega}_z$ in the body-rate controller. This is stopped once the motor velocity commands saturate during peak angular velocities rises (in post-processing of the flight log). The input constraints are then defined by the inequality

$$\underline{f}_i \leq f_i \leq \overline{f}_i \tag{5.23a}$$

$$\underline{\omega}_{xy,i} \leq \omega_{xy,i} \leq \overline{\omega}_{xy,i} \tag{5.23b}$$

$$\underline{\omega}_{z,i} \leq \omega_{z,i} \leq \overline{\omega}_{z,i} \tag{5.23c}$$

where, because of symmetry, the minimum angular velocity is defined as $\underline{\omega} = -\overline{\omega}$ for all axes. The full formation control optimization problem therefore becomes

$$\min_{\mathbf{u}_i} \quad \mathcal{O}_{\beta i}\left(\mathbf{x}_i(0), \hat{\mathbf{x}}_j(t), \mathbf{u}_i(t)\right) + \mathcal{O}_{\mathfrak{M}i}\left(\mathbf{x}_i(0), \mathbf{u}_i(t)\right) + \mathcal{O}_{\mathbf{u}i}\left(\mathbf{x}_i(0), \mathbf{u}_i(t)\right) \tag{5.24a}$$

$$\text{s.t.} \quad \underline{\mathbf{u}}_i \leq \mathbf{u}_i \leq \overline{\mathbf{u}}_i \tag{5.24b}$$

which may be solved using the optimization techniques discussed in the previous section. Not that the formation control optimization problem is formulated, we move on to the evaluation of its effectiveness as a closed-loop control method.

5.3 Simulations

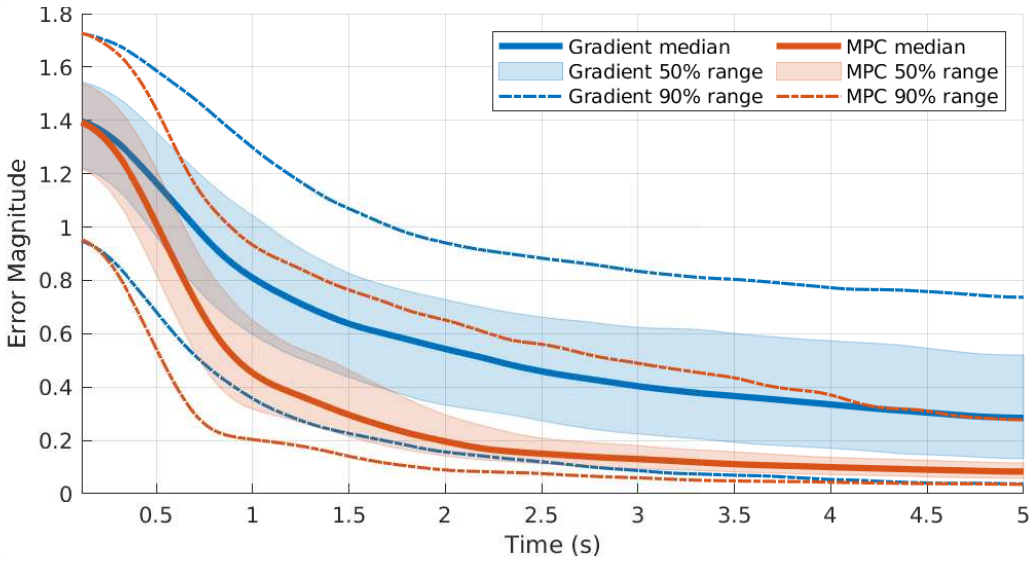
In this section, we present the simulation results performed for tuning the controller and for confirming the convergence properties of the controller which have not been theoretically

proven. First an extensive set of Matlab and Simulink simulations are performed to confirm that the control method has interesting convergence properties. Then more representative (with respect to our experimental platform) simulations using PX4 software-in-the-loop (SITL, see appendix A) and Gazebo are used to show that the control method can work well with reasonable prediction horizons and time steps.

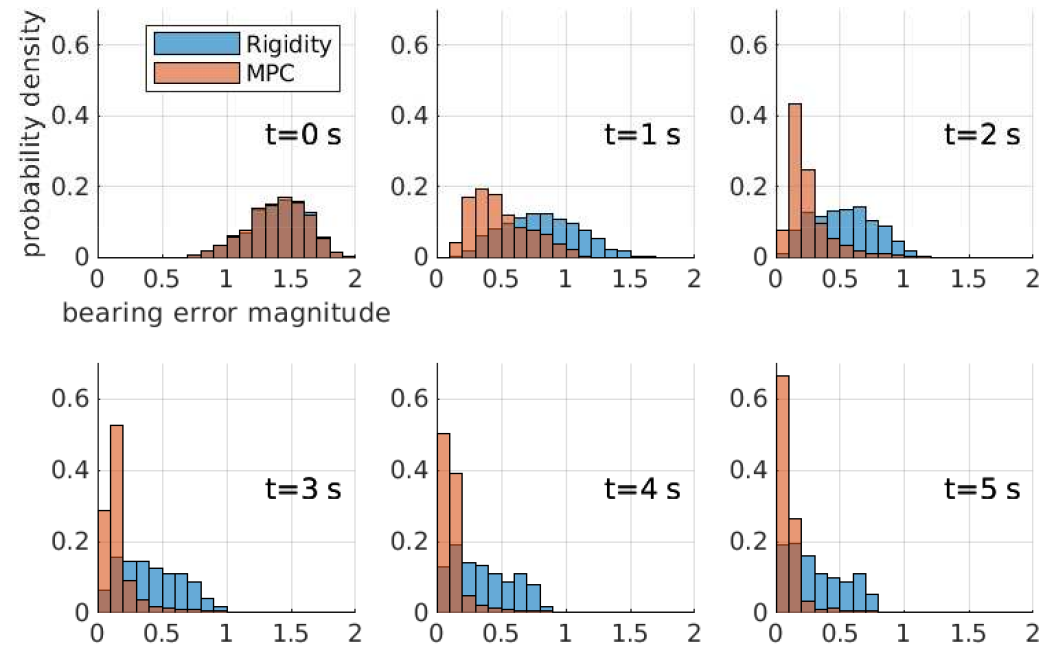
5.3.1 Formation convergence

As the decentralized finite-horizon MPC approach lacks the rigorous proof of convergence of rigidity methods, requires some approximations such as the estimated inter-agent distance, and is much more computationally expensive, it is important to first check that it will give some performance benefit. An improvement could manifest as a faster response for either the formation control or steering tasks, or as an increased basin of convergence (as multi-agent and visual servoing controllers are both well known to present local minima [179], [180]). Because the response of the systems may vary significantly between different initial and desired configurations, we decided to evaluate the MPC controller and rigidity control by running a large number of simulations. To this end, simulations were performed using Matlab and Simulink, where each agent was modelled by equations 5.9(a-d), with added first order propeller dynamics and random gaussian noise on the state feedback, bearing measurements, and propeller accelerations. The optimization was performed using the well-known *fmincon* function with an interior-point solver and numerically calculated gradients. The inner control loops of the UAVs and the forward dynamics were simulated at 200 Hz, and the bearing formation controller simulated at 30 Hz. Note that the numerical calculation of the gradient resulted in a simulated real-time factor of around 0.02, which would clearly be insufficient for real flights.

The results of 1100 simulations are presented in Fig. 5.3, comparing the MPC controller against the gradient-based rigidity controller. Only the results of simulations where at $t = 5$ s the mean bearing error is $||\mathbf{e}_\beta|| \leq 0.77$ (less that $45°$) are shown, as others are assumed to have fallen into local minima or become unstable, and make the results more difficult to interpret. This results in 12% of the flights using the rigidity controller and 4% of flights using the MPC controller being rejected. It can be seen that among the best flights, the steady state error was similar between the two controllers. The MPC controller however has a significantly faster transient response, and furthermore the grouping of experiments



(a) A continuous time box and whiskey plot for the bearing error of the simulation



(b) The probability density histograms for the normalized bearing error at snapshots throughout the simulations.

Figure 5.3 – The result of extensive simulations comparing the rigidity and MPC controllers

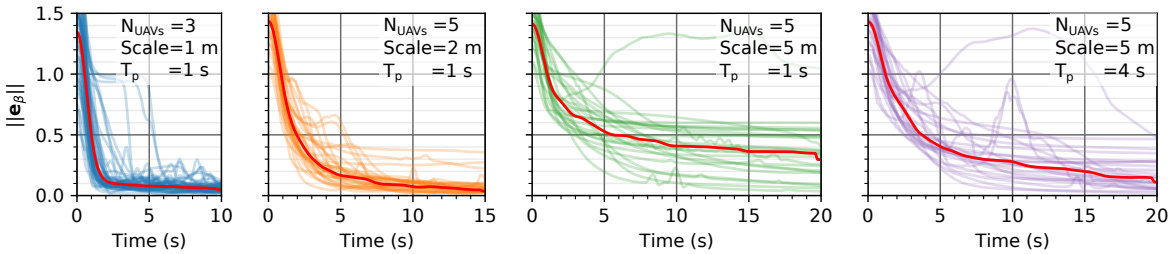


Figure 5.4 – Simulations assessing the required prediction horizon.

at steady state is much closer than for the rigidity controller. Indeed we can see that from $t = 1.75$ s and onwards, 75% of all flights using the MPC controller had errors within the range of the top 25% of flights using the rigidity controller. The results are quite convincing therefore that MPC is a beneficial control strategy and merits further study as applied to bearing formation control.

5.3.2 Tuning

Along with an evaluation of the formation convergence properties of the controller, we also use simulations to adjust the tuning parameters for the MPC. Unlike the previous section where Simulink was used to automatically run a large number of different simulations, we use the more physically and computational representative Gazebo simulator with PX4 SITL to tune the controller. Note that this controller uses the software implementation described further on in section 5.4and which is identical to the low-level and high-level software used in experiments. In the MPC formation controller, there are two main types of tuning variables to be chosen: prediction parameters and optimization function gains. Both need to be well chosen to produce a successful real-time controller, but have different effects. The optimization gains are used to define a point along the pareto-frontier between various objectives of the optimization, balancing the prioritization between tracking the desired trivial motions, reducing bearing error, and minimizing control effort. After some manual testing in simulation, the bearing control gain was chosen such that $\mathbf{Q}_{\beta i} = 75$ was the most important task, but that the trivial motions with a gain of $\mathbf{Q}_{\mathfrak{M}i} = 25$ is also quite important. The control effort gain of $\mathbf{Q}_{\mathfrak{M}i} = 5$ is sufficient to make the quadrotor tend towards hovering when all other criteria is satisfied, without impeding transient convergence properties for either the bearing error or the trivial motions.

The other parameter tuned in simulation was the prediction horizon. This was first tested

in simulation without considering onboard calculation times (discussed later in section 6.4). The prediction horizon of an MPC control should be similar in length to the time constant of the system, which vary significantly between formation configurations, but for a median simulated response in Fig. 5.3 it seems to be near to 1 s. Those simulations however all used a formation scale of 1 m, and it seems reasonable to assume that a larger formation would have a different time response to a change in formation shape. We tested three different formations by flying them through a sequence of 30 random desired shapes:

1. A three-UAV formation at a desired scale of 1 m
2. A five-UAV formation at a desired scale of 2 m
3. A five-UAV formation at a desired scale of 5 m

In Fig. 5.4, the results are shown for each configuration using a prediction horizon of 1 s. As we observe that the third configuration has relatively poor convergence properties, we test it again with the same control gains, but with a prediction horizon of 4 s. This improved both the transient response, and decreased the median steady state error from $||\mathbf{e}_{\beta}|| = 0.4$ to $||\mathbf{e}_{\beta}|| = 0.15$.

Because of the limited size of our flight arena, all formations must be kept fairly small, furthermore long prediction horizons add significant computational load. We therefore use a prediction horizon of $T_p = 1.5$ s for basic MPC formulations (as in this section), and of $T_p = 1.0$ s for more complicated optimization problems as is done in section 5.6, to remain within the required computation time limits. A time step of $\Delta t = 0.1$ s is used in all implementations, as it is similar to the rise time of the control variable ω.

5.4 Experiments

This section presents the experimental results obtained from the bearing formation MPC formulation presented up to this point. More complex implementations with path constraints are presented later, but this section evaluates the results of the baseline bearing formation MPC in Eq. (5.24) which is functionally equivalent to most other bearing formation controllers.

5.4.1 Real-Time Implementation

While simulations show the validity, and indeed the advantages of the MPC method, the results must be validated experimentally to show that our solution works well with existing technology. While an effort was made to accurately simulate the physical behaviour of the quadrotors (i.e. accurate dynamic models with noise, uncertainty and delays), it is more difficult to simulate the computational and data transfer characteristics of the drones. With an average run time of over four minutes for a single five second simulation (of a 3-drone formation), the *fmincon* optimization function is clearly non-feasible in real time. The main reason that the matlab simulations run below real speed is the optimization solver uses an inefficient numerically calculated gradient, and the inefficient single shooting prediction routine. To be able to solve this problem in real time, we need a solver:

1. Written in a fast executing language, such as C or C++

2. With pre-calculated gradients, using either code-generation or algebraic solutions to avoid computationally expensive numerical alternatives.

Without considering commercial MPC software designed for industrial applications, there are several open-source packages with successful real-time implementations. After considering the matlab MPC Toolbox (used in [35], [169]), the ETH Control Toolbox (used in [159], [181]), and NLOPT (used in [167], [182], [183]), we settled on the ACADO toolbox[1] (see [151], [184] for details) as our optimization solver, which is used in [168], [173], [185]. Developed for use in MPC, this toolbox allows the high-level formulation of the optimization problem in C++, and generates efficient C code for a SQP (sequential quadratic programming) optimization solver with multiple shooting discretization, RK4 integration, and using the QPoases library for solving sparse quadratic programming problems.

The ACADO-generated code is compiled with fixed state and objective dimensions, whereas we wished to employ variable numbers of bearing measurements. Rather than pre-compile separate code for each agent, we generated code based on a maximum number of observations per agent (we used four) and set the weights of unused measurements to zero at runtime. After trial and error, it was found that the best results were obtained when running the controller at 50 Hz, performing an RTI step of the SQP algorithm (about 5-8 ms

1. Note that ACADO (http://acado.github.io/) is now at the end of it's life. Future implementations may want to consider using its successor *acados* (https://docs.acados.org) instead.

per step) each loop. We used a prediction horizon of $T_p = 1.5$ s discretized into $N_p = 15$ steps, as discussed in the previous section.

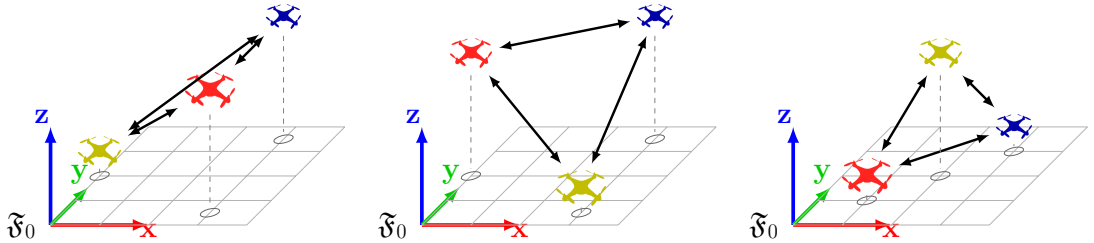
5.4.2 Experimental Results

The MPC method was tested in real-time experiments[2] on the platform presented in appendix A. All experiments began with the quadrotors hovering at pre-specified positions, and the formation controller takes over at time $t = 0$ s as seen in figures 5.5-5.6. As the experiments are performed in a small (relative to the formation footprint) $4 \times 6 \times 4$ m flying arena, an autopilot algorithm running at 20 Hz on the ground station laptop provides the $v_{\mathcal{F}x}$, $v_{\mathcal{F}z}$, and $\dot{s}_{\mathcal{F}}$ trivial motions in the frame of the formation to keep it centred along the short axes of the arena, and at a scale of 1 m. The other trivial motions $v_{\mathcal{F}y}$ and $\dot{\psi}_{\mathcal{F}}$ are given manually at 50 Hz using a joystick connected to the groundstation.

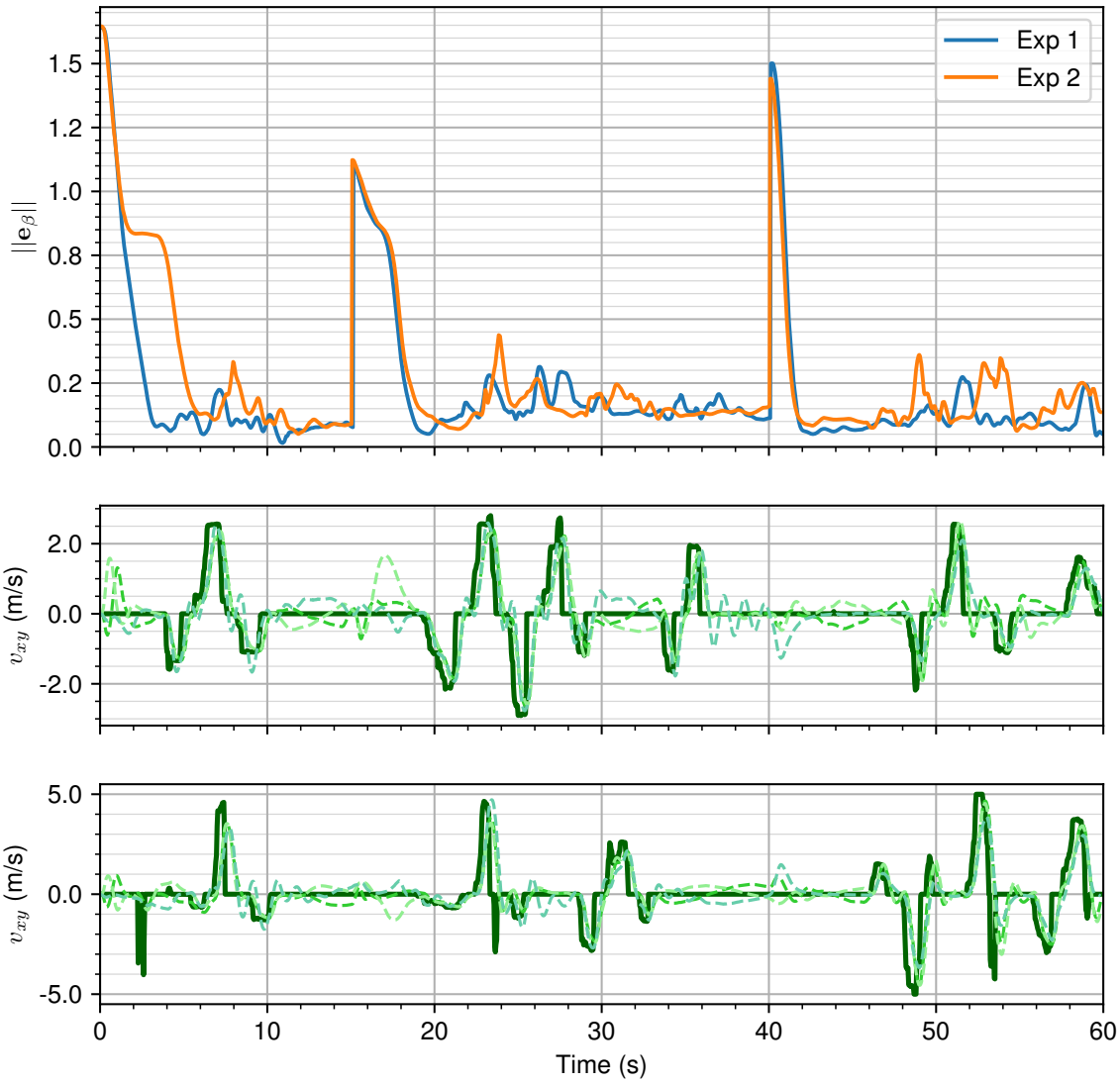
Two experiments each are shown for the three (E3.1-2) and four (E4.1-2) agent formations. We see in figures 5.5-5.6 that despite different pilot inputs, the experiments generally share very similar convergence properties. The difference in E3.1-2 for the initial formation convergence shows that due to slight differences in the initial configuration, the two tests followed different convergence paths but were still successful in attaining the desired shape. We can furthermore see that the maximum steering velocity (2.5 ms^{-1} in E3.1 and 5.0 ms^{-1} in E3.2) had some effect on the bearing control, as the peak bearing errors for E3.2 at 8 s, 24 s, 49 s and 52 s occur directly after the peak velocity times. This is possibly due to propeller saturation that occurs when attempting to re-orient the quadrotor while maintaining high thrust to reach accelerations of $\dot{v}_i > 12$ ms^{-2}, but also due to the nature of the multi-objective optimization, which must chose a balance between accelerating the UAVs to the desired velocity and accelerating them to maintain the desired shape.

The second pair of experiments show that similar results may be obtained when scaling up the formation and using variable numbers of edges. Experiments E4.1-2 had a one agent with a single bearing measurement, two agents with two bearing measurements, and a single agent with three bearing measurements. The steering was less aggressive (with peak velocities of 2.5 ms^{-1} and peak accelerations of 6 ms^{-2}) as the formation occupied more volume making it harder to pilot in the arena, and there was more aerodynamic interference

2. Videos of the experiments presented in this section may be found at https://box.ec-nantes.fr/index.php/s/FfrWWrwLaTfHoF5

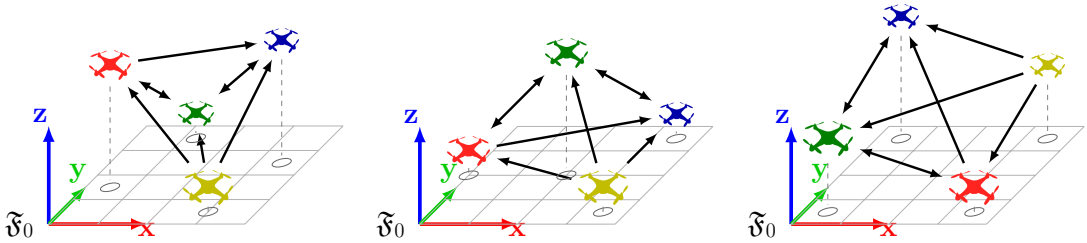


(a) Formation #1 (b) Formation #2 (c) Formation #3

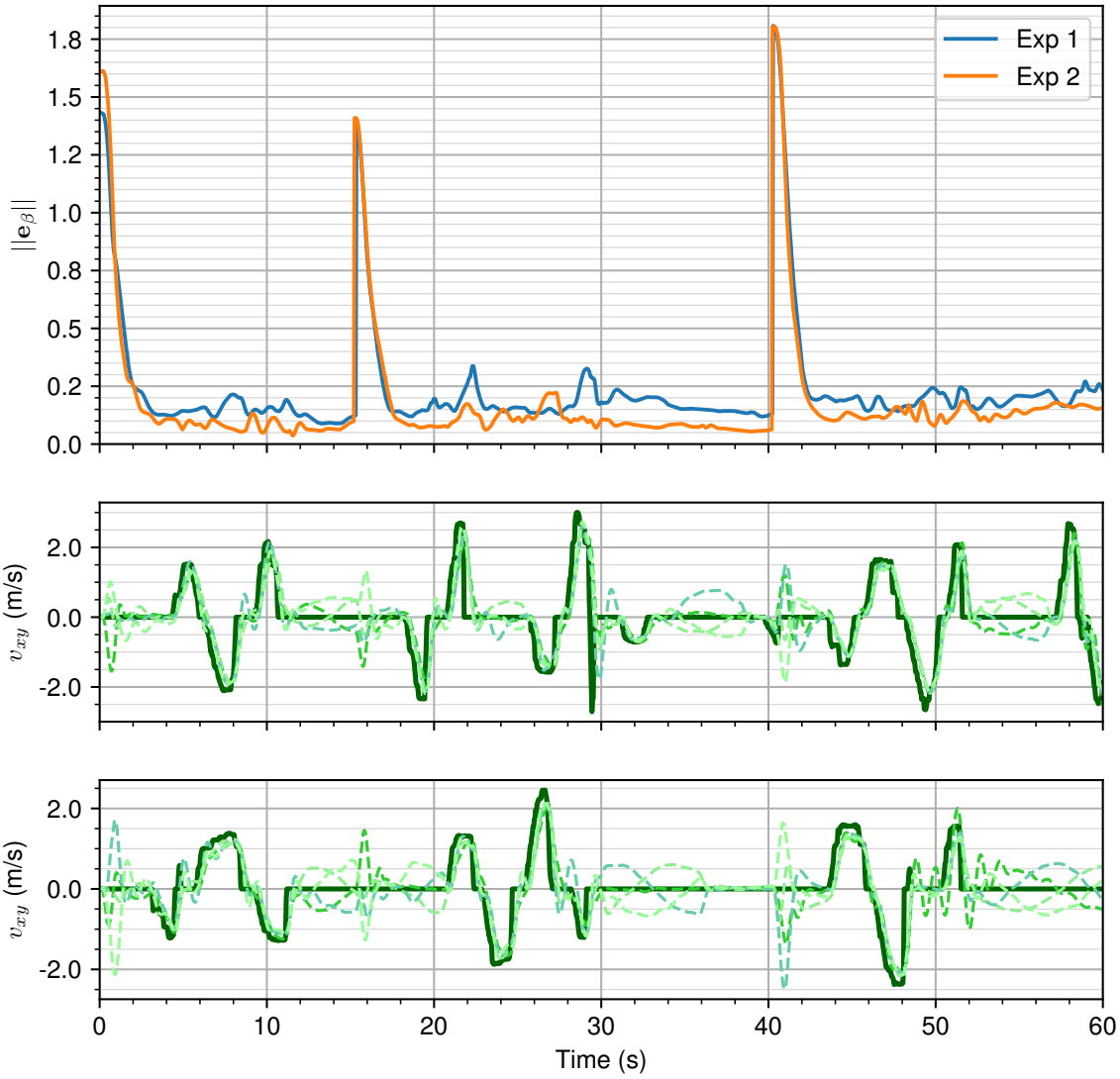


(d) Bearing error and trivial motion tracking results

Figure 5.5 – The top plot shows the evolution of the normalized bearing error for E3.1 (blue) and E3.2 (orange). The middle and bottom plots shows the desired formation velocity $\mathbf{v}^d_{\mathcal{F},y}$ (dark green line) and the measured velocities $\mathbf{v}_{i,y}$ (light green lines) expressed in $\mathfrak{F}_0$ for E3.1 and E3.2. respectively. The bearing error peaks correspond to the desired formations in (a)-(c).



(a) Formation #1 (b) Formation #2 (c) Formation #3



(d) Bearing error and trivial motion tracking results

Figure 5.6 – The top plot shows the evolution of the normalized bearing error for E4.1 (blue) and E4.2 (orange). The middle and bottom plots shows $\mathbf{v}^d_{\mathcal{F},y}$ (dark green line) and the measured $\mathbf{v}_{i,y}$ (light green lines) expressed in $\mathfrak{F}_0$ for E4.1 and E4.2. respectively. The bearing error peaks correspond to the desired formations in (a)-(c).

between quadrotors. We can see in Fig. 5.6 the yaw rate of the formation (e.g. 30-36 s in E4.2 among others) visible due to the distinctive sine wave induced by $\omega_{z\mathcal{F}}$.

5.5 Extended MPC - Disturbance Rejection

It can been seen in our experiments that despite a good transient bearing convergence, the steady state bearing error is quite high. Experiments in [7] show that the rigidity control method can bring the bearing error to effectively zero, while the flights in our experiments showed steady state bearing errors in the range of 0.08 to 0.20 (in the absence of aggressive trivial motions). Because of the heavy filtering, the presence of integral control in the lower-level control, and the slow trivial motions used in [7], it is unlikely that the MPC method will perform as well as steady state, however it is still desirable to minimize the error as much as possible. The primary reason for the poor steady state performance is likely the unmodelled effects that are not accounted for in the optimization prediction. Some examples include

1. Actuation uncertainties such as
   (a) approximate curve fitting for the actuation model (see the thrust identification in appendix A)
   (b) battery characteristics which change with age, use, temperature
   (c) modified aerodynamics due to damaged propeller blades, air temperature, etc...
2. Quadrotor mass uncertainties, as some batteries had masses of ± 25 g around the nominal mass used in the MPC model.
3. External aerodynamic forces, for example
   (a) The downdraught and turbulence from other UAVs. This is likely the dominant disturbance in our experiments
   (b) The ground effect force when flying near the ground
   (c) Environmental air flows such as wind, convection currents, etc...

While some of these could be incorporated into the model by tighter experimental protocols and improved modelling of the UAV actuation systems, the lack of disturbance rejection is a strong limitation of the MPC method.

In order to overcome this issue, we modify the optimization model to include an estimation of the unmodelled forces acting on the quadrotor. This will be supplied from an onboard observer based of the assumption that the control thrust is accurately generated and that the mass is accurately known, but that an unknown disturbance force acts on the quadrotor. The linear acceleration component of the prediction model then becomes

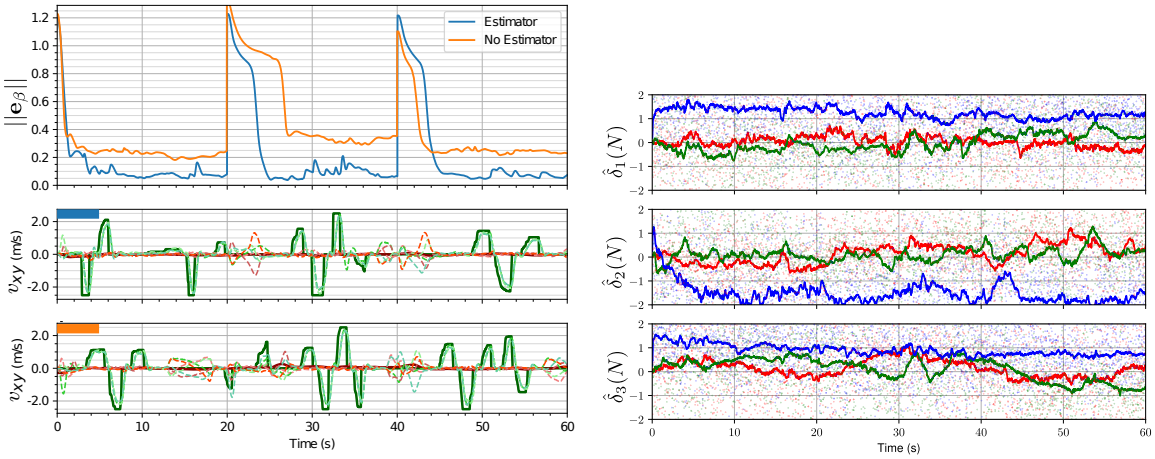
$$\dot{\mathbf{v}} = \left(\frac{f_i}{m_i} \mathbf{e}_3 - \mathbf{R}_i^T \mathbf{g} + \frac{\boldsymbol{\delta}_i}{m_i} \right) \tag{5.25}$$

where $\boldsymbol{\delta}_i \in \mathbb{R}^3$ is the unmodelled force vector acting on $\mathcal{A}_i$ and expressed in $\mathfrak{F}_i$. As the acceleration $\dot{\mathbf{v}}_i$ of $\mathfrak{F}_i$ may be measured using the IMU, the disturbance may be estimated as a first order system

$$\dot{\hat{\boldsymbol{\delta}}}_i = k_e \left(\hat{\boldsymbol{\delta}}_i - m_i(\dot{\mathbf{v}}_i + \mathbf{R}_i^T \mathbf{g}) + f_i \mathbf{e}_3 \right) \quad \text{where } k_e > 0 \tag{5.26}$$

which from an initial estimate of $\hat{\boldsymbol{\delta}}_i(0) = \mathbf{0}$, will converge to the unknown disturbance force. This can be included in the MPC control prediction step replacing Eq. (5.9)b and reprojecting the disturbance estimate $\hat{\delta}_i$ onto the $\mathbb{R}^3 \times \mathbb{S}^1$ manifold, as well as in the desired equilibrium inputs by setting the desired hover thrust in Eq. (5.21) to $f_i^d = ||m_i \mathbf{g} + \hat{\boldsymbol{\delta}}_i||$.

In a first experiment to confirm that this method works correctly, a 3-UAV formation is flown in the flight arena at speeds of 2 ms^{-1} and cycling through three different desired formation shapes. Quadrotor 1 underestimates its mass by 200 g (17%), quadrotor 2 under-produces thrust, and quadrotor 3 over-produces thrust. It can be seen in Fig. 5.7a that without active compensation, the modelling errors create significant bearing errors, indicating that the formation shape is poorly controlled. With an estimator however the bearing error is reduced to a magnitude as low as 0.05. There is indeed a stronger correlation between manoeuvring accelerations and the bearing error with the disturbance estimator active (likely due to the simple nature of the estimator used, a more performant observer might improve upon this) however this does not bring the error even to the lowest level of static error without the estimator. One can see the effect of the disturbance observer in Fig. 5.7, with the blue lines (the estimated forces along the $\mathbf{z}_0$) showing the mismatches with the nominal model. As the mass of quadrotor 1 is overestimated, the quadrotor requires less thrust than expected to produce a given acceleration. This is represented in the observer (under hovering conditions) by a disturbance force pushing up on the UAV. Likewise and



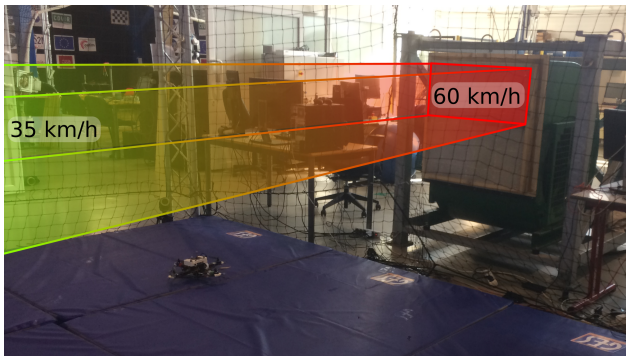
(a) The bearing error with and without disturbance estimation for uncertain robot mass

(b) The estimate of the disturbance for acting of each UAV for the experiment with estimation

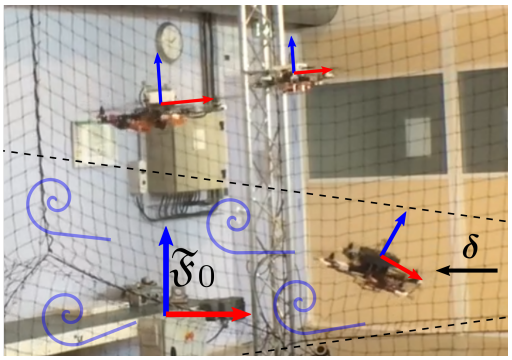
Figure 5.7 – The bearing results with and without disturbance estimation for model uncertainty. In both experiments we use a three-UAV formation, with one UAV over-estimating it's mass by about 200 g, and the other two under and over produce thrust.

quadrotor 3 over-produces thrust, the observer interprets this as producing the correct amount of thrust plus a positive disturbance in the thrust direction. Quadrotor 2 under-produces thrust, and therefore the observer estimated that there is a disturbance pushing down on the UAV, and thus the controller adds thrust to compensate.

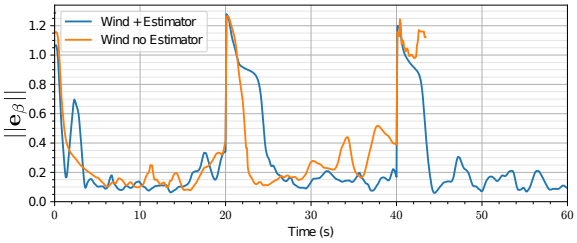
Even if the quadrotors are correctly modelled however, there may still be environmental disturbances such as turbulence, wind-gusts, down-draughts, etc... We therefore evaluate the success of the observer-compensated MPC controller using a fan which generates horizontal winds in the arena, measured from 60 km/h (17 m/s) close to the fan to 35 km/h (10 m/s) far from the fan, along a fairly narrow beam as shown in Fig. 5.8a. Note however that as a wall is directly on the other side of the flight arena, the airflow is not entirely predictable, with significant turbulence and reflection. This added disturbances along the $\mathbf{x}_0$ axis of over 4 N (the estimated disturbances in $\mathfrak{F}_0$ can be seen in Fig. 5.8d), as well as some disturbances in the $\mathbf{z}_0$ direction. The formation was flow through a series of shapes, while passing back and forth through the beam of wind, and the bearing error is shown in Fig. 5.8c. With the estimator, the bearing error was generally kept below 0.2, while it was higher for tests that did not include the estimator in the MPC prediction. The performance of the later tests were difficult to quantify however, as purely from a piloting perspective, if was difficult to fly through the wind without crashing. UAVs in the wind beam using the estimator tended to quickly compensate for the disturbance as seen in Fig. 5.8d and keep their approximate position,



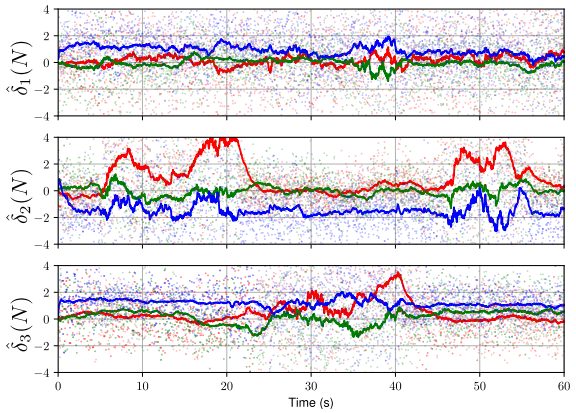
(a) An approximate representation of the disturbance distribution in the arena



(b) A UAV with disturbance estimator compensating for a cross-wind disturbance



(c) The bearing error with and without disturbance estimation for an experiment with 3 UAVs



(d) The disturbance estimates of for each UAV expressed in $\mathfrak{F}_0$ for the experiment in (c) making use of the estimator in the MPC

Figure 5.8 – The bearing results with and without disturbance estimation, for perturbations due to uncertain mass and wind.

however without the estimator they were almost directly blown into the net surrounding the flight arena.

5.6 Extended MPC - Constrained Bearing Formations

So far, the formation controllers have only been tasked with minimizing the objective function subject to the actuation constraints of the quadrotors. In some cases however it may be interesting to go a step further and apply more complex constraints that must be respected in real applications. We have identified several key constraints that may be of practical interest, in order to evaluate their effect on the performance of formation MPC, although this is not by any means an exhaustive list:

1. Respect altitude limits to avoid collisions with objects or humans

2. Avoid collisions with external objects for which the position relative to the UAVs are known

3. Maintain all the observed UAVs in the field of view of the onboard cameras

4. Avoid collisions with neighbouring UAVs in the formation

Because we are dealing with a real system with noisy measurements and estimated states, we cannot guarantee that a constraints will never be violated. If a constraint is violated at measurement time, the optimization algorithm may not be able to find any feasible solution that respects all constraints, and will produce effectively arbitrary control inputs which cause the UAV to fly in an uncontrolled and dangerous manner. We therefore make use of soft constraints, whereby the real constraints may be violated at the expense of a large penalty on the objective function [167].

Some of the listed constraints are somewhat antithetical to the principle of bearing-only control as information pertaining to relative distance for neighbouring UAV collision is taken as at best a roughly estimated value for the controller. For external obstacles however, if we assume that these are large compared to UAVs, it becomes possible to measure the relative distance with a rangefinder, RGB-D, or stereo camera anti-collision system, as there is a large surface area to detect. The constraints therefore take more liberties with regards to the evaluation of exact relative positioning than do the formation control objectives. We develop and test the first three constraint types, although collision constraints with neighbouring UAVs can easily be implemented as well, so long as a reasonably accurate distance estimate could be obtained.

5.6.1 Altitude constraints

Altitude constraints are very simple and yet practical requirements for flight envelops, and will serve here as a demonstration of the soft constraint formulation. Recalling the generic formulation of the optimization algorithm in Eq. (5.3), the minimum and maximum altitudes ($\underline{p}_z$ and $\bar{p}_z$) can be expressed as two inequalities

$$\mathbf{p}_i^T \mathbf{e}_3 > \underline{p}_z \tag{5.27a}$$

$$\mathbf{p}_i^T \mathbf{e}_3 < \bar{p}_z \tag{5.27b}$$

where the position is not really necessary, instead we could use only the altitude of the UAV, which can be estimated easily with a barometer and/or downward-facing range finder. These two constraints could be rigorously enforced, assuming that the initial position is inside the feasible set (and that the prediction horizon is sufficiently long to stop the upward or downward velocity of the UAV). If however the quadrotor is at the constraint boundary and either an unmodelled force such as a downdraught forces the quadrotor below $\underline{p}_z$, or a noisy measurement places the quadrotor's estimated altitude below $\underline{p}_z$, there may be insufficient thrust available to make the first prediction step $\mathbf{p}_i(\Delta t)^T\mathbf{e}_3 > \underline{p}_z$ in which case there is no feasible solution to the optimization problem. We recall however that the control inputs are by definition chosen to satisfy the constraints while minimizing the objective function. Soft constraints therefore create a new control input ϵ_{alt} called a "slack variable" such that

$$0 \leq \epsilon_{\text{alt}} \leq \bar{\epsilon}_{\text{alt}} \tag{5.28a}$$

$$0 \leq \mathbf{p}_i^T\mathbf{e}_3 - \underline{p}_z + \epsilon_{\text{alt}} \tag{5.28b}$$

$$0 \leq \bar{p}_z - \mathbf{p}_i^T\mathbf{e}_3 + \epsilon_{\text{alt}} \tag{5.28c}$$

where ϵ_{alt} is a control variable that can be arbitrarily assigned by the optimization so as to satisfy inequality Eq. (5.28). By penalizing the use of this control value in the objective function by introducing the quadratic objective term

$$\mathcal{O}_\epsilon = \boldsymbol{\epsilon}^T\mathbf{Q}_\epsilon\boldsymbol{\epsilon} \tag{5.29}$$

where $\boldsymbol{\epsilon}$ is the vector of all slack constraint variables and $\mathbf{Q}_\epsilon$ is the positive diagonal weighting matrix for the slack constraints. This allows for a violation of the constraints in Eq. (5.27) by penalizing the square of the magnitude of the violation. The soft constraint gains should be quite large compared to the objective gains to "stiffen" the constraint so as to limit the magnitude of violation. Note that also that if an upper bound is given to the slack variable such that $\epsilon_{\text{alt}} < \bar{\epsilon}_{\text{alt}}$, then when this value is reached the constraint becomes a hard constraint and violations are not feasible solutions. The objective function penalty as a function of altitude for a limited soft constraint on the minimum and maximum altitude is shown in Fig. 5.9, where the altitudes shaded in red will be infeasible solutions, and the altitudes in blue will be penalized.

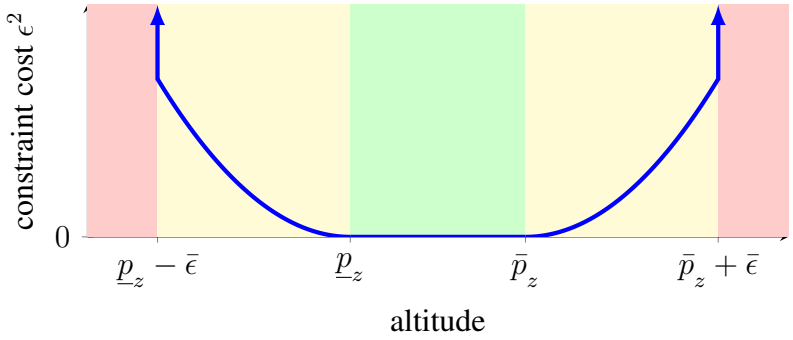
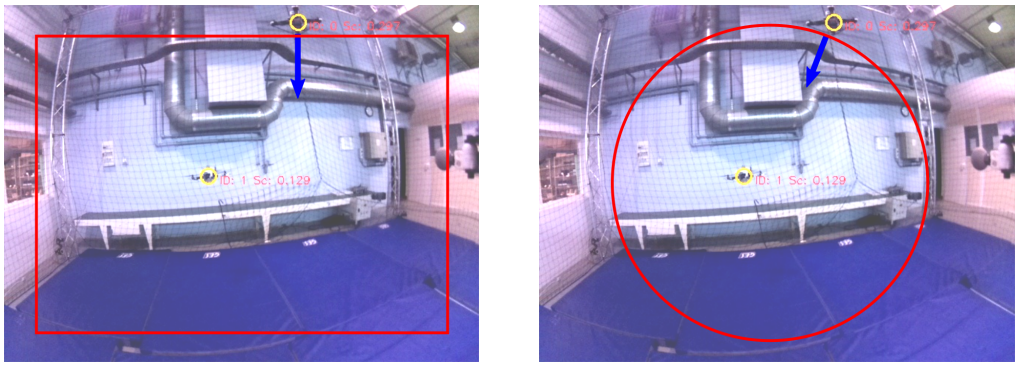


Figure 5.9 – The cost of a soft altitude constraint. The red zones are infeasible solutions to the optimization problem, the yellow zones are penalized by a slack variable $0 < \epsilon \leq \bar{\epsilon}$, and the green zone is unaffected by the constraint, thus $\epsilon = 0$

5.6.2 Field of View Constraints

While some UAVs have full omnidirectional camera coverage, this increases the cost of the UAV, the mass, and the computing required for image processing. It is therefore common for UAVs to be equipped with only a single camera, either rigidly fixed to $\mathfrak{F}_i$ or mounted on a motorized gimbal with a known rotation relative to $\mathfrak{F}_i$. Individual cameras often have a field of view (FOV) less than $180°$ with the measurement being distorted and less accurate near the edges, and with no information available beyond the border of the FOV. It is of course desirable that all desired inter-robot bearings be measurable, and thus we will create soft constraints to endeavour to keep the observed UAVs inside the observing UAV's FOV. There are several methods including constraining the distance from the rectangular borders of the image (see Fig. 5.10a), or by considering the radial distance from the center of the image (as in Fig. 5.10b). We will consider the later as it avoids moving into pixel-space allowing us to deal only with bearings, and because it prevents the use of the corners of the image where distortion and inaccuracy are greatest (at the expense of losing some of the measurement space, limiting the set of achievable formation shapes). The radial FOV method has the additional advantage of requiring only a single slack variable to enforce the FOV, whereas a rectangular constraint would require two slack variables to behave efficiently, as the distance from the horizontal and vertical image borders are independent.

Let us begin by assuming that the principle axis of the camera (see appendix B for details) is aligned with the $\mathbf{z}_c$ axis of the camera frame $\mathfrak{F}_c$ and passes through the center of the image. If the camera is mounted with a rotation ${}^i\mathbf{R}_c$ relative to $\mathfrak{F}_i$, the principle axis of the camera may be found as $\mathbf{z}_c = {}^i\mathbf{R}_c\mathbf{e}_3$. As cameras often come with angular field of view specifications, we choose to define the distance of the bearing measurement from the



(a) Rectangular FOV constraint (b) Radial FOV constraint

Figure 5.10 – The action (blue arrow) of two type of FOV constraints.

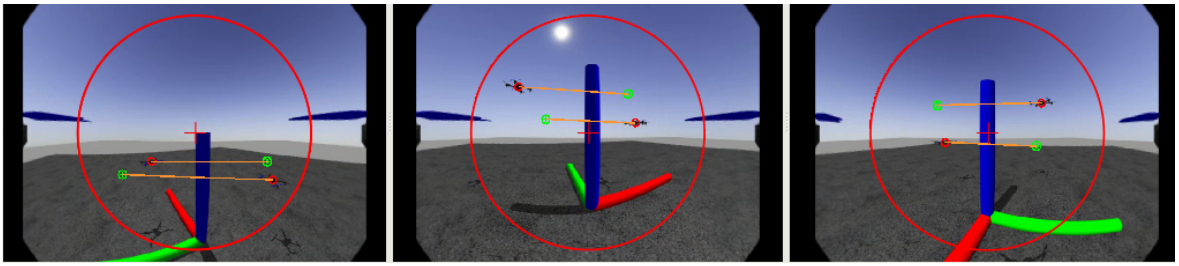


Figure 5.11 – The steady state view from UAVs 1, 2, and 3 at a local minima (in a Gazebo simulation). The green points mark the desired bearings and the red points the measured bearings. The red circle is the boundary for the activation of the FOV constraint penalization.

principle axis by the minimum FOV angle α_{FOV}. As the bearing is projected onto a spherical manifold, the difference between two intersecting bearings spanning an angle α is $\boldsymbol{\beta}_1 - \boldsymbol{\beta}_2 = 2\sin(\alpha/2)$. By considering $\mathbf{z}_c$ to be a bearing expressed in $\mathfrak{F}_i$, the FOV constraint on a bearing $\boldsymbol{\beta}_{ij}$ can therefore be expressed as a function of the camera FOV, which leading to the slack inequalities

$$0 \leq \epsilon_{FOVij} \tag{5.30a}$$

$$0 \leq \epsilon_{FOVij} + 2\sin(\alpha_{FOV}/4) - ||\boldsymbol{\beta}_{ij} - \mathbf{z}_c|| \tag{5.30b}$$

where α_{FOV} is the lesser of the horizontal and vertical FOV values. This soft constraint will not necessarily prevent neighbouring UAVs from leaving the FOV, however it will penalize inputs which cause the observed UAVs to go beyond the the circle defined in image space by the limiting FOV angle. It is therefore recommended to reduce α_{FOV} below the real value, as the penalizing action in the control only begins once this value is exceeded. Care must also be taken to ensure the desired formation bearings respect these constraints, otherwise the soft constraint will induce a bearing error even at steady state.

The addition of FOV constraints will also have the undesired effect of introducing local minima into the control problem as is defined up to now. This is because the penalty for violating a soft constraint will be greater than the benefit of reducing the bearing objectives, and thus the formation will be stuck in the local minima which reduces the bearing error as much as possible without violating the soft constraints. This generally occurs when a UAV must cross the formation, causing the observed UAVs to switch sides in the image, and is manifested by horizontal parallel lines joining $\boldsymbol{\beta}_{ij}$ and $\boldsymbol{\beta}_{ij}^{d}$ as shown in Fig. 5.11. To avoid becoming stuck in a local minima, a high terminal cost on the bearing error

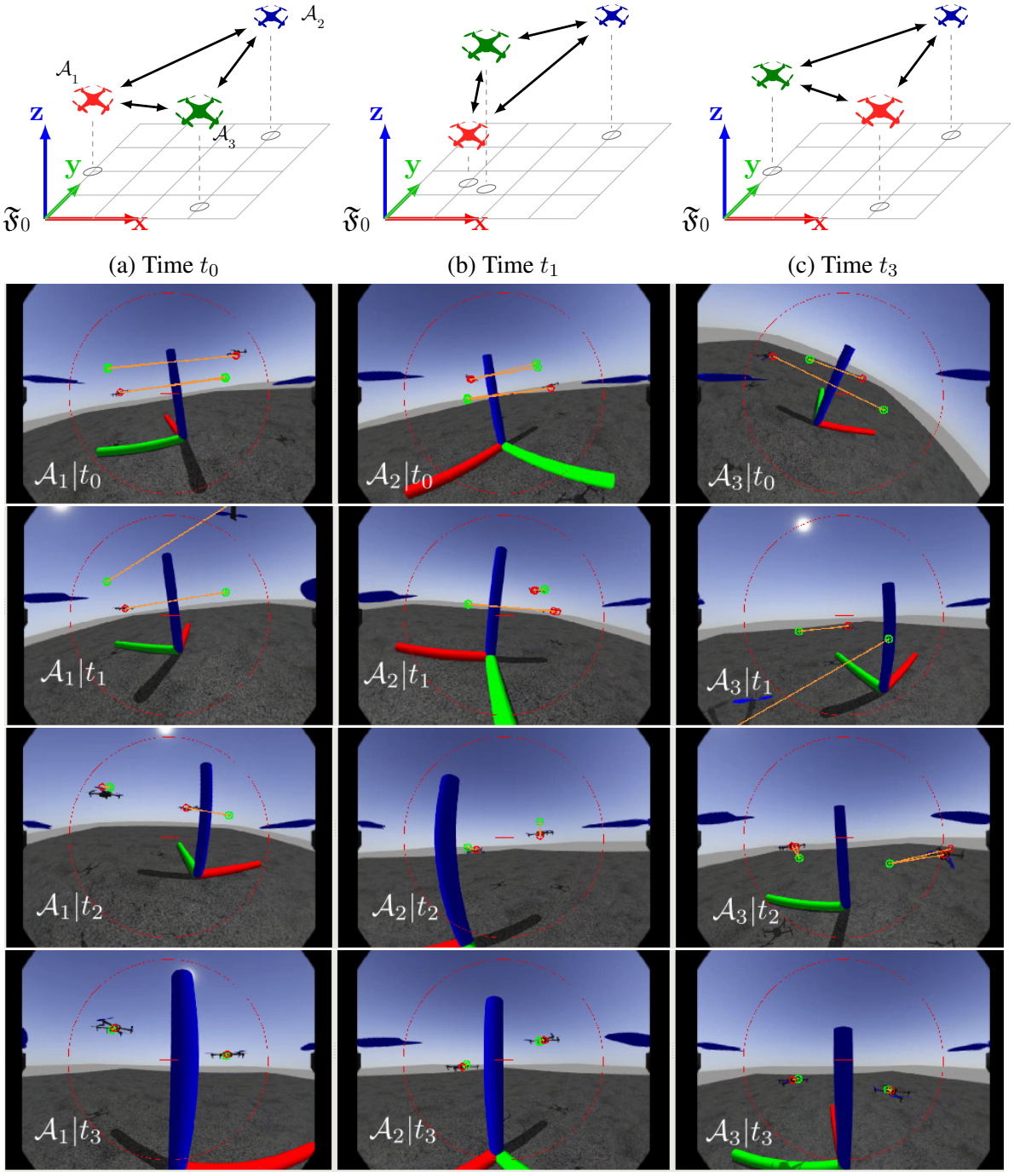
$$\mathcal{O}_{\beta N} = \mathbf{e}_{\beta}^{T}(T_p)\mathbf{Q}_{\beta N}\mathbf{e}_{\beta}(T_p) \tag{5.31}$$

is added so that the optimization will find that is in fact beneficial to briefly violate some constraints in order to minimize the bearing error at the end of the prediction horizon. This terminal cost is successful in escaping from local minima, however it is remarked that in all tested cases, at least one UAV had at least one of its bearing measurements leave the FOV as shown in the example in Fig. 5.12. The interest however in enforcing FOV constraints is due to the fact that outside the FOV the bearing may not be measured, and in such a case it would not be coherent to use these measurements in the controller. In experiments where measurements may be sporadically unavailable, the Kalman filter and its variations are generally seen as the de-facto solution. In our experiments considering FOV constraints such as Fig. 5.12 and 5.13, we only measure $\boldsymbol{\beta}_{ij}$ when $\mathcal{A}_j$ is truly in the camera image of $\mathcal{A}_i$ (apart from a measurement at the first control iteration, regardless of whether it is detected or not). For example in Fig. 5.12, between t_1 and t_2, the bearings $\boldsymbol{\beta}_{13}$ and $\boldsymbol{\beta}_{31}$ pass out of the image frames and are only estimated by the Kalman filter process without measurements being taken. More on this may be found in appendix B.

The effect of using FOV constraints and the importance of terminal costs is demonstrated using experiments with real-time onboard vision with a deep learning-based UAV detection algorithm. The formation (a three-agent complete graph) is tested in three different scenarios, with the desired bearings switching between random feasible formation shapes[3] every 15 s. The scenarios tested are:

1) Without FOV constraints or terminal costs ($k_{\text{FOV}} = 0$, $k_{\beta N} = 0$). In this scenario, the soft constraint gains are set to zero. It thus has no effect on the control, despite the slack

3. The random seed is the same for each scenario and thus the sequence of desired formations are the same.



(a) Time t_0 (b) Time t_1 (c) Time t_3

Figure 5.12 – View from UAVs $\mathcal{A}_1$, $\mathcal{A}_2$ and $\mathcal{A}_3$ (columns) at four snapshots in time $t_0 \ldots t_3$, (rows) when converging to a new desired formation. The green points mark the desired bearings and the red points the measured bearings. The red circle is the boundary for the activation of the FOV constraint penalization. Note that at t_1, agents $\mathcal{A}_1$ and $\mathcal{A}_3$ cross over each other, leaving each other's FOV, re-entering before t_2. A video comparing the effect on of terminal cost on the local minima in FOV-constrained formation controllers can be found at https://box.ec-nantes.fr/index.php/s/4b5pXJ4ZxCMkiTm.

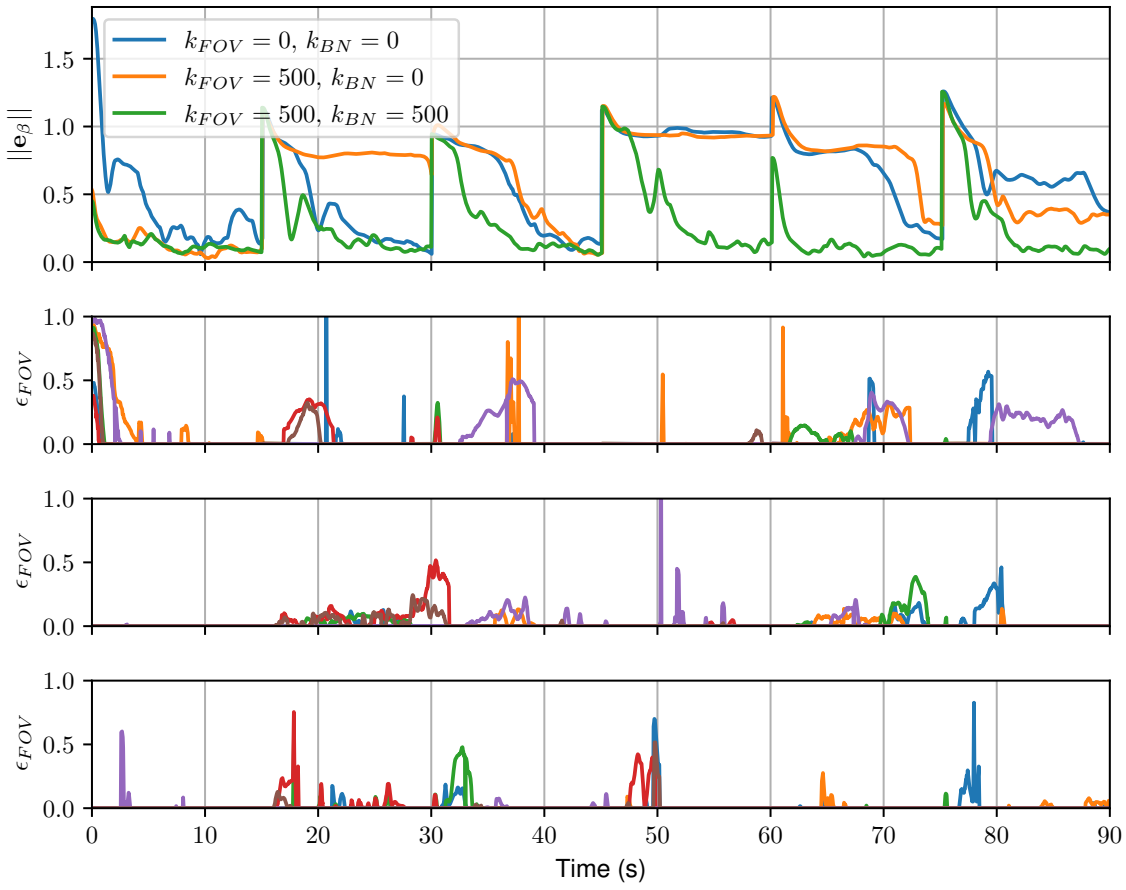


Figure 5.13 – Real-time experimental results with FOV constraints and onboard bearing detection. The upper plot shows the bearing error for the three flights, and the lower plots shows the (six) slack variables for the three flights respectively.

Table 5.1 – A quantitative evaluation of FOV constraints. The total activation is the time integral of the slack variables, and the mean activation is the total activation averaged across the time active. The time active (when $\epsilon > 0$) has units in seconds.

	Time $\epsilon > 0$	Total activation	Mean activation	Mean $\|\mathbf{e}_{\boldsymbol{\beta}}\|$
$k_{\text{FOV}} = 0 \quad , k_{\boldsymbol{\beta}N} = 0$	35.0	9.1	0.26	0.61
$k_{\text{FOV}} = 500, k_{\boldsymbol{\beta}N} = 0$	45.9	4.5	**0.10**	0.63
$k_{\text{FOV}} = 500, k_{\boldsymbol{\beta}N} = 500$	**18.2**	**2.4**	0.13	**0.26**

variable still being evaluated.

2) With FOV constraints (the soft constraint gain is $k_{\text{FOV}} = 500$) and without terminal costs ($k_{\boldsymbol{\beta}N} = 0$).

3) With FOV constraints ($k_{\text{FOV}} = 500$) and terminal cost ($k_{\boldsymbol{\beta}N} = 500$).[4]

For each scenario the formation is manually piloted along the $\mathbf{y}_0$ axis of the arena at speeds of $||v_{\mathcal{F}_y}|| \leq 2.5$ ms^{-1} as in previous experiments and the bearing error may be seen in the upper plot of Fig. 5.13. The lower three plots show the values of the six FOV slack variables (two constrained bearings per UAV) for the three flights respectively.

It can be clearly seen that that terminal cost is beneficial to ensuring convergence, even in the absence of constraints. However when FOV constraints are included, the importance of the terminal cost increases, as the controller requires a stronger impulsion to pass through the local minima introduced by the soft constraints. The effectiveness of the FOV constraints can be seen in Fig. 5.13, where the slack variables are much less active in the third scenario with FOV constraints and terminal costs than in the other flights. In particular, while soft constraint violations do occur, they are generally brief (0-2 s) occurrences. When there are no FOV constraints, there are often extended periods (5-8 s) of time when the FOV is not respected. Table 5.1 shows the total slack variable activation time, the sum of the integrations of the slack variables over the entire flight (total activation) and that value divided by the activation time (mean activation). The latter gives an indication of on average how far outside the target FOV area the measurements are when they are outside. Note that while the mean activation is lower without a terminal cost, this is because rather than quickly crossing infeasible FOV states, the local minimum states lie on the border of the FOV with low magnitude constraint violations (see 15-30 s and 60-75 s), but at a high bearing error.

In this thesis, we consider only cameras rigidly mounted to the quadrotor frame, however it is common with larger UAVs for the cameras to be mounted on a motorized gimbal as in

4. The video of this experiment may be found at https://box.ec-nantes.fr/index.php/s/KxBHEJLatQgDJdG

[170]. In such a case the roll and pitch of the camera with respect to $\mathfrak{F}_i$ (denoted θ_c and ϕ_c) would become control variables and the objective term could be designed which reduces the mean distance of each bearing from the center of the camera

$$\mathcal{O}_c = k_c e_c^2 \quad \text{where } e_c = \sum_{j=1}^{\mathbb{N}_i} ||\mathbf{R}_y(\theta_c)\mathbf{R}_x(\phi_c)\mathbf{e}_3 - \boldsymbol{\beta}_{ij}|| \tag{5.32}$$

This objective function would effectively reduce the value of e_c so that the only constraint on keeping all measured UAVs in the camera FOV is the maximum angular separation between the observed UAVs. In such a case, both local minima and FOV constraint violations may often be completely removed.

5.6.3 Obstacle Collision Constraints

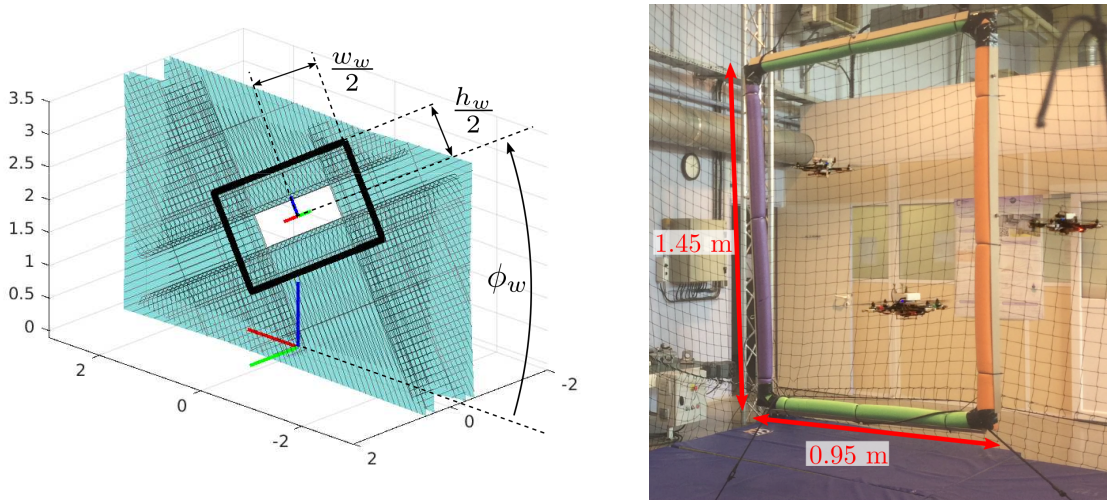
Avoiding collisions with obstacles are a classical type of MPC constraint, and are used to enforce a minimum distance between the robot and the contact surface of a physical object or of an area that is to be denied to the robot. We begin by defining the collision surface of the quadrotor as a sphere of radius r_i centred on $\mathfrak{F}_i$, and which includes some margin of safety beyond the propeller tips. This minimum distance must be enforced with respect to all obstacles, although as described earlier, the constraint is soft and therefore some small amount of penetration within the sphere will occur before the controller forces the UAV away from the contact. The most basic 3D obstacle to avoid is another sphere of radius r_s, and at position $\mathbf{p}_s$ relative to $\mathfrak{F}_i$. The slack inequality that must be satisfied may therefore be expressed as

$$0 \leq \epsilon_s \leq \bar{\epsilon}_s \tag{5.33a}$$

$$0 \leq \epsilon_s + \sqrt{\mathbf{p}_s^T \mathbf{p}_s} - (r_i + r_s) \tag{5.33b}$$

where $\bar{\epsilon}_s$ is the maximum allowable overlap of the robot sphere and the obstacle sphere.

There are many ways to model collision boundaries, but often for the sake of simplicity they are modelled as an approximation by regular shapes such as cylinders, spheres, and polygons. In these experiments we will represent an infinite wall with a rectangular window who's center is positioned at $\mathbf{p}_w$ relative to $\mathfrak{F}_0$, with height h_w, width w_w and a skew angle ϕ_w as represented in Fig. 5.14a. Some works consider windows too small to pass through without



(a) The constraint surface (blue) for an wall bisecting the world along $\mathbf{x}_0$ with a small window.

(b) View of the obstacles in a Gazebo simulation.

Figure 5.14 – Environmental constraints considered in this work

high dynamics, using either a dual trajectory planning and trajectory tracking approach [159] or by modelling the boundary of the quadrotor as a set of ellipsoids [161] resulting in a very large number of constraints to be solved rather than as a sphere. In this work we simply consider the wall as a single obstacle and the UAV collision surface as a single sphere, thus the window must be large enough for a UAV to pass through at any orientation. We will begin by assuming that each UAV is able to estimate its position relative to the frame on the window $\mathfrak{F}_w$ (see [183], [186] for work on gap detection and relative pose estimation for UAVs), which is positioned at the center of the window, and oriented such that the x-axis $\mathbf{x}_w$ is orthogonal to the plane of the window and that $\mathbf{y}_w$ and $\mathbf{z}_w$ intersect and are orthogonal to two adjacent window edges. The distance of $\mathcal{A}_i$ from the wall is therefore the x-component of its position relative to the window $x_i = {}^w\mathbf{p}_i^T\mathbf{e}_1$ expressed in $\mathfrak{F}_w$, and can be used to easily construct polynomial constraints

$$
\begin{aligned}
{}^w\mathbf{p}_i^T\mathbf{e}_2 &\leq \quad k_w x_i^4 + \frac{w_w}{2} - r_i - \epsilon_{w,y} && \text{(5.34a)}\\
{}^w\mathbf{p}_i^T\mathbf{e}_2 &\geq -k_w x_i^4 - \frac{w_w}{2} + r_i + \epsilon_{w,y} && \text{(5.34b)}\\
{}^w\mathbf{p}_i^T\mathbf{e}_3 &\leq \quad k_w x_i^4 + \frac{h_w}{2} - r_i - \epsilon_{w,z} && \text{(5.34c)}\\
{}^w\mathbf{p}_i^T\mathbf{e}_3 &\geq -k_w x_i^4 - \frac{h_w}{2} + r_i + \epsilon_{w,z} && \text{(5.34d)}
\end{aligned}
$$

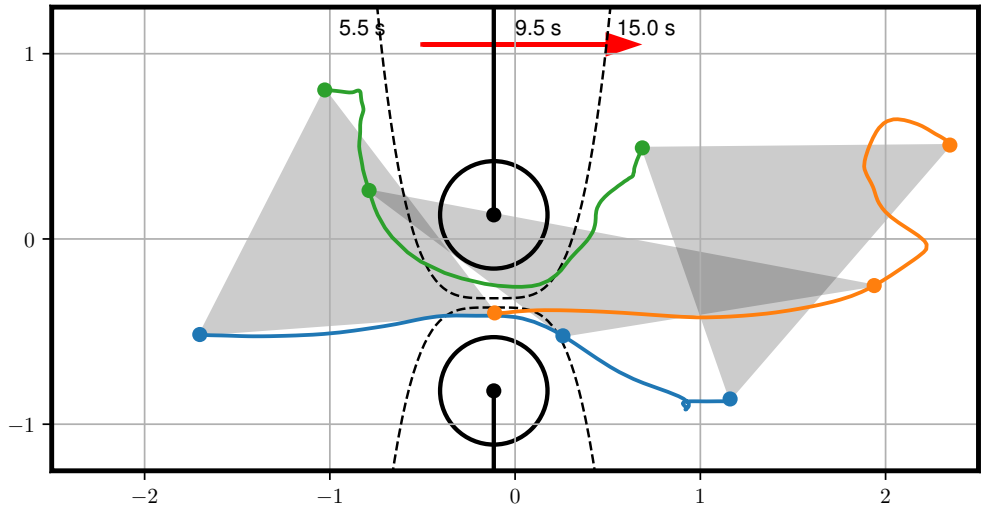
which make use of two slack variables, $\epsilon_{w,y}$ and $\epsilon_{w,z}$, and the steepness of the polynomial

is set by k_w. A polynomial representation was chosen because it is convex, assisting convergence of the optimization algorithm, and has a width, which prevents the UAV from jumping from one side to the other between two discrete prediction time steps. These polynomial constraints may be seen in 3D in Fig. 5.14a, and are tested in experiments as is shown in Fig. 5.14b. Note that in these experiments, the FOV constraints were disabled because the combination of FOV constraints and collision constraints resulted in lengthy RTI computation times, which caused the UAVs to become unstable (see section 6.4 for details).

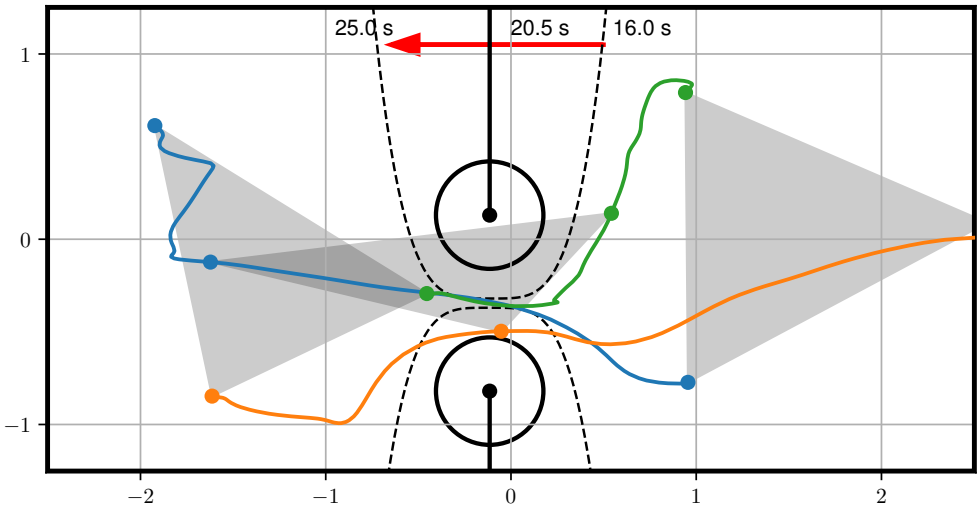
To test the environmental obstacle avoidance, we flew a 3-UAV formation in the flight arena, which was bisected by an invisible virtual wall. The formations was piloted from one side of the arena to the other, and could only cross the wall at the window, located approximately in the center. Each UAV has access to their position relative to $\mathfrak{F}_w$ using the MOCAP system, although in practice there are existing onboard vision-based estimators that could be used in the absence of MOCAP. The window has a height of 1.45 m, a width of 0.95 m, and is held approximately vertical. Each UAV has a center to blade tip radius of 29 cm, but a nominal radius of $r_i = 45$ cm was included in the constraint formulation to give a buffer for the soft variables to act, the gains of which were set to $k_{Obs} = 750$. A constraint steepness of $k_w = 10$ m^{-3} was used, and to help encourage the UAVs to pass through the window rather than stopping on the constraint border, a terminal cost of $k_{\mathfrak{m}N} = 200$ on the velocity component of the nullspace objective was implemented.

The results of one of these experiments can be seen in Fig. 5.15, where the formation is flown back and forth through the window[5]. Figures 5.15a-b show the top view for a left to right pass, and the return right to left pass. The two center dots represent the locations of the vertical window borders and the circles around each window border represent the collision radius of the quadrotor (i.e. the path of the quadrotor touching the circle would mean the blade tip of a propeller would touch the window). The constraints of each window (Eqs. 5.34a-b with $\epsilon_{w,y} = 0$) are represented by dashed lines, and the path of each quadrotor by a coloured line. Three snapshots are taken in each figure at the times indicated, with the gray triangle representing the shape of the formation (the desired shape from a top view is an equilateral triangle). While a skilled pilot may be able to manoeuvre to formation through the window in the desired shape using multiple different trivial motions, here the formation is simply piloted with a left-to-right (Fig. 5.15a) or right-to-left (Fig. 5.15b)

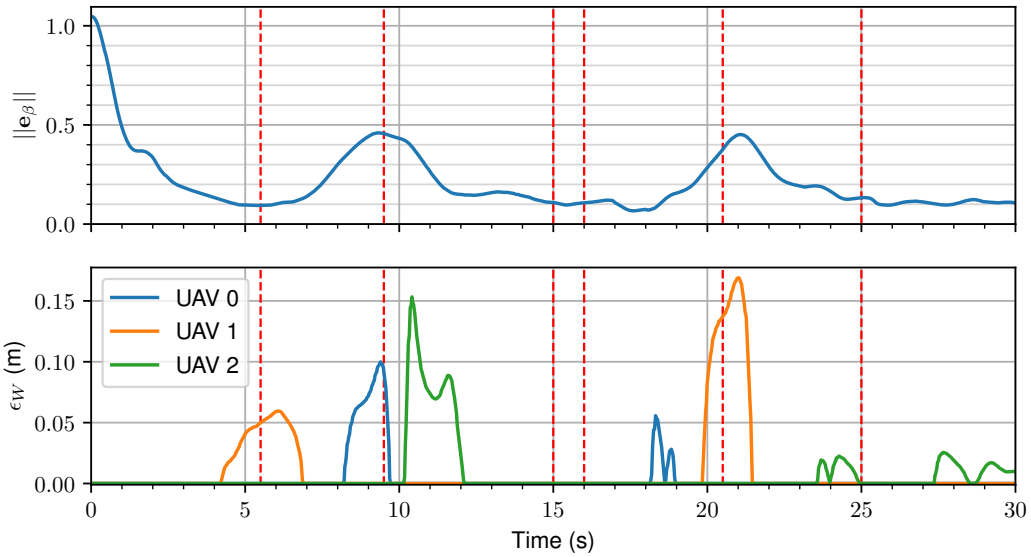
5. A separate experiment may by seen at https://box.ec-nantes.fr/index.php/s/Z3woYNJfD6XEyLG



(a) The first pass through the window (left to right). See page 125 for details.



(b) The second pass through the window (right to left). See page 125 for details.



(c) The normalized bearing error (top) and slack variable $\epsilon_{w,y}$ of each UAV (bottom). The red lines represent the time of each snapshot in (a) and (b)

Figure 5.15 – Experimental results flying through a window in formation.

velocity. The decentralized controllers thus attempt as best they can to compromise between maintaining the formation shape, the desired formation velocity, and avoiding collisions with the environment. This results in the formation deforming (see the increased bearing error in Fig. 5.15c as the slack variables become active) and rotating as it passes through the window, and restoring itself to the desired shape once the UAVs are sufficiently far from obstacles. It can be seen that twice a quadrotor comes very close to colliding with the window (the green quadrotor in Fig. 5.15a at t=10.5 s and the orange in Fig. 5.15b at t=21 s), and thus the gain on the slack variable or the constraint radius r_i should be increased in the constraints for a safer margin of error.

5.7 Conclusions

In this chapter we showed that MPC can be used to calculate a low-level body-frame angular velocity and thrust control signal that is often successful in solving the bearing formation control problem. The low-level action of the control variables allows for high dynamics, at the expense of increased steady state error variability relative to the SOVS experiments, most of which can nonetheless be reduced by the use of simple disturbance estimators. The MPC control problem can be formulated with varying degrees of complexity depending on required task and on the computational resources available, and both simple and complex versions were validated in realistic simulations and in real-time experiments.

Simple

$$\min_{\mathbf{u}_i} \quad \mathcal{O}\left(\mathbf{x}_i(0), \hat{\mathbf{x}}_j(t), \mathbf{u}_i(t)\right) \tag{5.35a}$$

$$\text{s.t.} \quad \underline{\mathbf{u}} \leq \mathbf{u} \leq \overline{\mathbf{u}} \tag{5.35b}$$

Complex

$$\min_{\mathbf{u}_i, \boldsymbol{\epsilon}} \quad \mathcal{O}\left(\mathbf{x}_i(0), \hat{\mathbf{x}}_j(t), \mathbf{u}_i(t), \hat{\boldsymbol{\delta}}_i(t)\right) \tag{5.36a}$$

$$\text{s.t.} \quad \underline{\mathbf{u}}_i \leq \mathbf{u}_i \leq \overline{\mathbf{u}}_i \tag{5.36b}$$

$$\mathbf{0} \leq \boldsymbol{\epsilon}_i \leq \bar{\boldsymbol{\epsilon}}_i \tag{5.36c}$$

$$\mathcal{C}_{\text{alt}} \geq 0 \tag{5.36d}$$

$$\mathcal{C}_{\text{FOV}} \geq 0 \tag{5.36e}$$

$$\mathcal{C}_{\text{obs}} \geq 0 \tag{5.36f}$$

While this work shows encouraging results for the successful application of MPC to realistic bearing formation control problems, there are of course open axes of research that merit further investigation. Recent developments in “Tube MPC" [187], [188] the formal

guarantees of interval of interval analysis by modelling the predictions as a gaussian process, potentially giving the MPC a more robust collision prevention model, however they are currently too computationally expensive for real-time deployment on onboard computers. We also only treated external disturbances as the action of an unknown force on the robot, which compensates for both external disturbances and intrinsic modelling errors through a lumped force observer. Currently there is rapid development in the combination of online machine learning (ML) and MPC [189]–[191], where ML is used to improve the model mismatches not only from external force but also from the inter-UAV distance, sensor bias and other issues. This in turn may provide better state estimations and predictions, leading to a more realistic optimization and thus better control results. Finally, improved objective functions and prediction methods could be designed to reduce the occurrence of local minima, to find more energy efficient methods of rearrangement, and to improve the decoupling between the bearing and manoeuvring objectives.

CHAPTER 6

COMPARISONS OF BEARING FORMATION CONTROLLERS

In the previous chapters, two new bearing formation controllers have been developed, which we believe will have more interesting dynamic properties than existing controllers. Initial experiments showed that both controllers can be successfully implemented on real UAV platforms, however they are significantly more complicated than the existing gradient-based methods generally used in bearing formation control. In this section we perform a detailed comparison between the newly developed SOVS and MPC controllers and the rigidity controller of [118] (the action of each controller can be seen in Fig. 6.1). Criteria include the reliability and response time for formation convergence, the dynamic performance when tracking various trajectories, the robustness to noise and modelling errors, and the implementability. Most data in this section is the result of extensive SITL Gazebo simulations in order to compare large numbers of flights in a controlled manner.

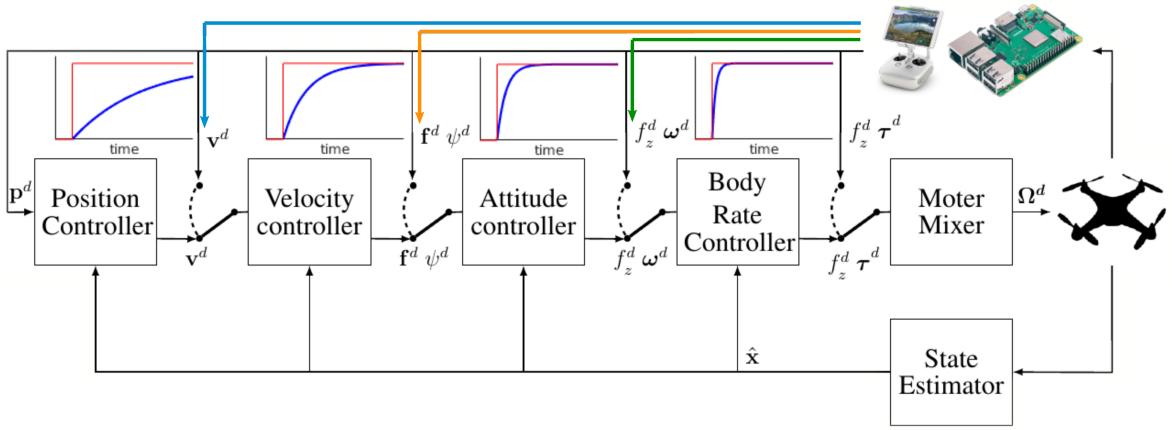


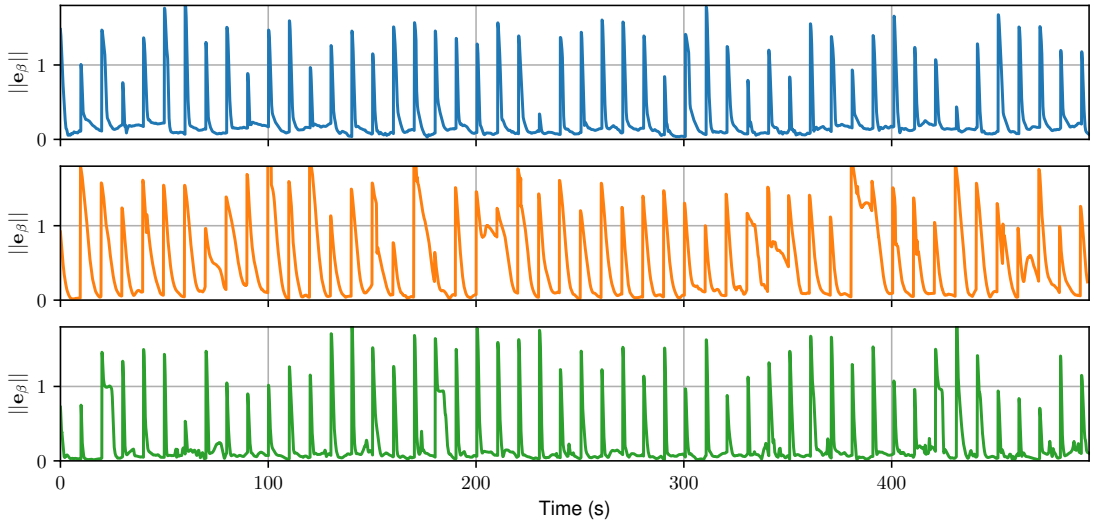
Figure 6.1 – A diagram demonstrating the flow of the three controllers considered. The rigidity controller (blue) provides a velocity command, the SOVS controller (orange) a thrust vector command and the MPC (green) a thrust and angular velocity command. A qualitative representation of the rise time of the low-level dynamics for each controller is shown.

6.1 Formation Convergence

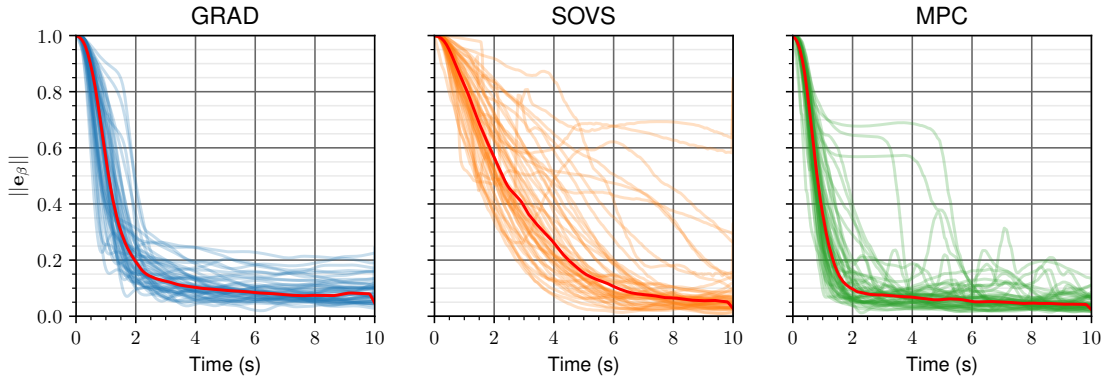
The first point of comparison is what may be considered as the principle objective of formation control, which is the ability to rearrange the robots such that they form the desired geometric shape defined by the set of desired bearings $\boldsymbol{\beta}_{ij}^d$. The three controllers[1] are tested in simulation by a sequence of random step changes to the desired bearings. This is performed twice, using a three-UAV formation and a five-UAV formation. An autopilot is used to generate trivial motions to keep the center of the formation at a given position and at a given scale (1 m for the 3-UAV formation and 2 m for the 5-UAV formation).

The results of the convergence tests for the 3-UAV formation are presented in Fig. 6.2. It can be seen that the rigidity controller and the MPC controller have similarly shaped responses, with MPC performing generally better as expected from the results of the previous chapter. When considering the rise time to 10% of the initial error, the median rigidity controller response was around 4.0 s, while the median MPC controller response was around 2.0 s. The MPC controller also demonstrated better steady state performance, with a median bearing error of 4% of the initial error over the 10^{th} second, while the median error with the rigidity controller was 8% over the same time span. While the exact decent properties depend on tuning parameters, it can also be seen that at steady state the MPC controller led to a tighter grouping of results, with only 3/50 tests being above 10% of the initial error, while 15/50 tests with the rigidity controller were above 10% error at that time. Nonetheless,

1. Unless otherwise stated, the MPC formation controllers in this chapter have no path constraints but includes disturbance estimation and rejection

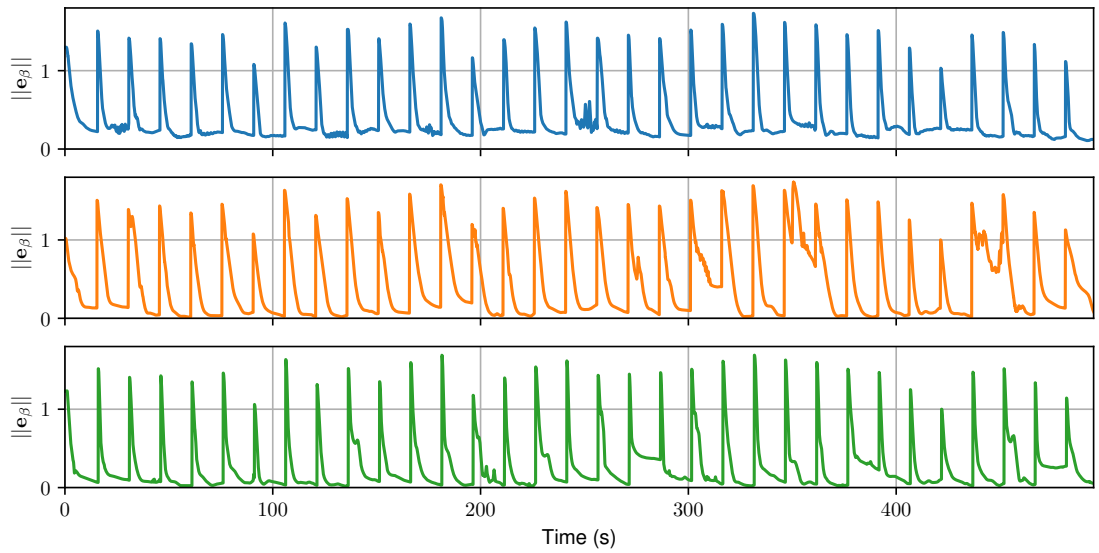


(a) The bearing error for 50 desired formations at intervals of 10 s with the rigidity controller (blue), SOVS controller (orange) and MPC controller (green).

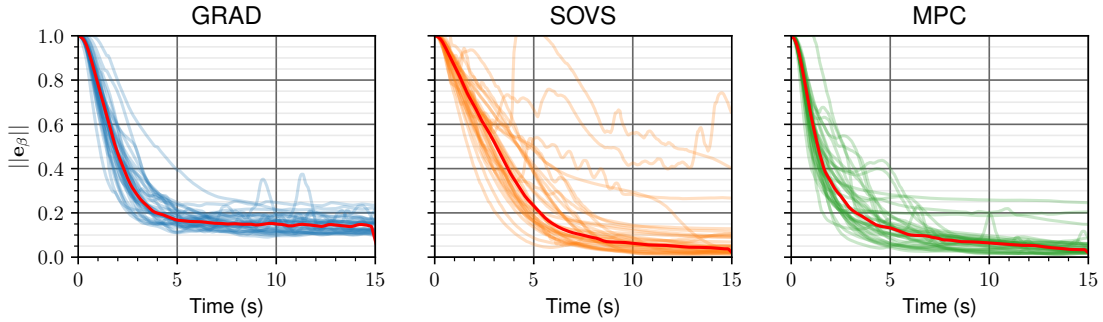


(b) The bearing error convergence as a fraction of the initial bearing error for each of the 50 changes in formation shape (median response in red).

Figure 6.2 – Bearing error convergence three-UAV formations



(a) The bearing error for 50 desired formations at intervals of 10 s with the rigidity controller (blue), SOVS controller (orange) and MPC controller (green).



(b) The bearing error convergence as a fraction of the initial bearing error for each of the 50 changes in formation shape (median response in red)

Figure 6.3 – Convergence experiments for a five-UAV formation

the rigidity controller has the advantage in that there are no outliers in transient performance, while in the MPC controller there are three tests which plateau at a large error for up to 5 s before converging. Looking now at the SOVS controller for the 3-UAV formation, it can be seen that the transient convergence is much poorer that for the other two controllers, with a median 10% rise time of 6.2 s. Nonetheless, its median steady state error over the 10^{th} second of convergence is close to that of the MPC controller at 5% of the initial value. While 10/50 tests had a final error greater that 10%, it can be seen from the plots that the transient performance is highly variable with a wide range of convergence profiles compared to the rigidity and MPC controllers, supporting the observations made in chapter 4 regards the difficulty in avoiding collisions with the flight arena netting during the transient phase of the controller.

The results of the five-UAV formation convergence tests are shown in Fig. 6.3 and was

run with the same control parameters as the three-UAV test, except that the desired formation scale was increased from 1.0 m to 2.0 m because of the increased number of UAVs. Because the inter-agent distance is inversely proportional to the control gain in the rigidity controller [118], the transient response of the rigidity controller is similar as before but approximately half the speed. A desired formation step interval of 15 s is thus used instead of 10 s due the slower convergence that will arise from the increased inter-robot distance. Indeed this is the case for the rigidity and MPC controller which both converge around half the speed, but the convergence rate of the SOVS is almost identical to the previous 3-UAV formation test. This is because the dynamic inversion of the SOVS controller allows the formation to track the performance of the damped second-order system specified by the auxiliary control law. In cases where the SOVS didn't converge (e.g. 345-360 s and 435-450 s) the thrust was saturated, although it is difficult to determine whether the failed convergence was due to the saturated thrust or vice-versa.

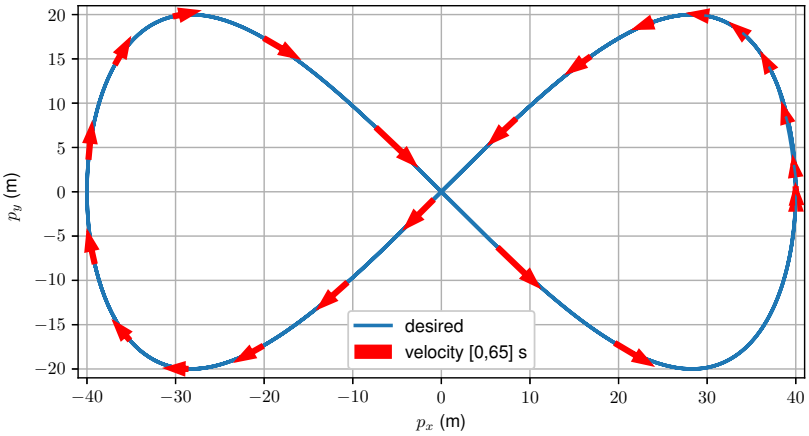
6.2 Dynamic Performance

Whereas the previous section only considered a formation hovering in a desired position and changing shape, in this section we consider a formation which attempts to maintain a constant desired shape while following a trajectory. Because this thesis does not have a specific use case, two scenarios are imagined:

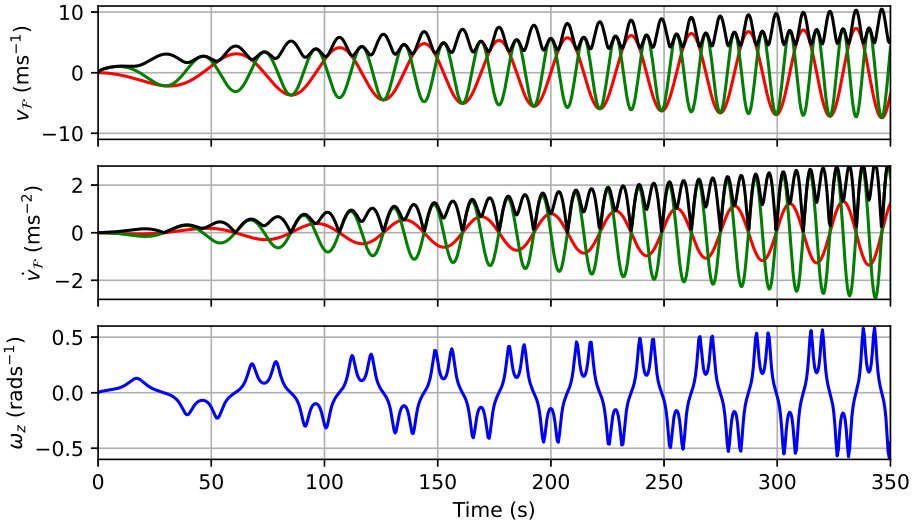
1. A planned trajectory with smooth positional input reference. This could be for a task such as boundary surveying and patrolling, or the inspection of infrastructure such a railways, pipelines and transmission lines with a known macroscopic geometry.

2. A reactive trajectory where the reference input may change at a given time, and may have discontinuous heading and accelerations. This could be the case in scenarios such as the pursuit of a non-communicating agent, deviating from a path to avoid an unplanned obstacle, or when following sharp-angled paths such as hallways as fast as possible.

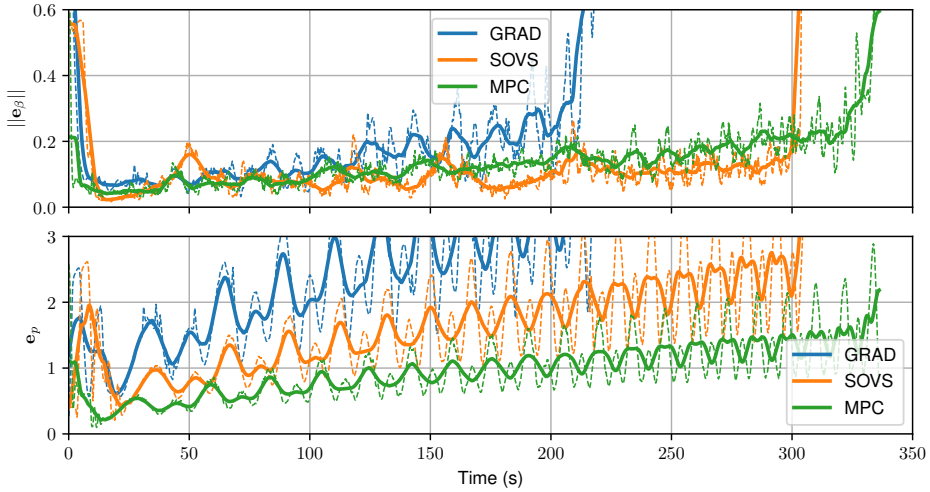
These two scenarios are used to compare the performance of the rigidity (as a baseline), SOVS, and MPC formation controllers using a 3-UAV complete graph formation.



(a) Velocity (proportional to the length of the arrows) and formation heading for the first period (0-70 s) of the trajectory



(b) Desired velocity, acceleration, and angular velocity profile. Red, green and blue are used for the x, y, and z axes, and black represents the magnitude.



(c) A plot of the bearing error and the trajectory tracking error for the three formation controllers. Solid lines are filtered and dashed lines are raw.

Figure 6.4 – Trajectory profile and results for the lemniscate trajectory

6.2.1 Smooth Trajectories

To test the performance of the formation controllers when performing smooth trajectories, we use an accelerating lemniscate trajectory (see Fig. 6.4a-b). The desired trajectory is defined in $\mathfrak{F}_0$ by

$$\mathbf{p}^d_{\mathcal{F}}(t) = \begin{bmatrix} A\cos(Pt^n) \\ \frac{A}{2}\sin(2Pt^n) \\ z^d_{\mathcal{F}} \end{bmatrix} \tag{6.1a}$$

$$\psi^d_{\mathcal{F}}(t) = \text{atan2}(\mathbf{e}_2^T\dot{\mathbf{p}}^d_{\mathcal{F}}, \mathbf{e}_1^T\dot{\mathbf{p}}^d_{\mathcal{F}}) \tag{6.1b}$$

where for these tests, $A = 40$ m, $P = 0.01$ s^{-n}, $n = 1.5$, and $z^d_{\mathcal{F}} = 5$ m. An eye-in-the-sky autopilot is assumed to have a knowledge of the position of the center of the formation $\mathbf{p}_{\mathcal{F}}$ and the heading of the formation $\psi_{\mathcal{F}}$ (defined by the axis $\mathbf{p}_1 - \mathbf{p}_{\mathcal{F}}$ projected onto the xy plane) in $\mathfrak{F}_0$, and sends steering commands $\mathfrak{M}$= [$\mathbf{v}_{\mathcal{F}}$ $\dot{\psi}_{\mathcal{F}}$ $\dot{s}_{\mathcal{F}}$] to each of the UAVs, which are computed using the proportional feed forward control law

$$\mathbf{v}^d_{\mathcal{F}} = k_p(\mathbf{p}^d_{\mathcal{F}} - \mathbf{p}_{\mathcal{F}}) + \dot{\mathbf{p}}^d_{\mathcal{F}} \tag{6.2a}$$

$$\dot{\psi}^d_{\mathcal{F}} = k_p(\psi^d_{\mathcal{F}} - \psi_{\mathcal{F}}) + \dot{\psi}^d_{\mathcal{F}} \tag{6.2b}$$

such that the center of the formation closely tracks the desired formation position and heading. The scale rate is regulated by a proportional control law based on the average distance of the UAVs to the formation COM. Note that while the trajectory may be computed for any point in time, we assume that the MPC controller is not aware of the future trajectory and reacts purely based on the autopilots current signal.

The results[2] of the smooth trajectory tracking are represented in Fig. 6.4c, with the top graph showing the bearing error and the bottom graph showing the desired position tracking error. With the rigidity controller, there was a significant trade-off between the amount of filtering, which improved the bearing error stability but reduced the position tracking performance. It can be seen that at approximately equivalent bearing tracking performance to the other controllers, the rigidity controller had quite poor position tracking, with error magnitudes fluctuating between 1 m and 5 m. The rigidity controller becomes unstable

2. Videos of the lemniscate simulations for the MPC and SOVS formation controllers may be seen at https://box.ec-nantes.fr/index.php/s/o2KGZZncPGdW6mk.

around 210 s. The SOVS and MPC controller have better performance in terms of bearing tracking error and stability, with the SOVS controller crashing around 300 s and the MPC controller around 340 s. The SOVS controller has a mean trajectory tracking error, however if the nullspace control gains in the SOVS controller are increased, the tracking error can be brought down to the level of the MPC controller. This however comes at the expense of greater instability in the bearing control task which becomes unstable earlier at around 200 s.

6.2.2 Non-smooth Trajectories

Non-smooth trajectories may be more indicative of potential reactive formation control tasks, and may be important if navigating in poor visibility conditions or with an unpredictable reference (e.g. in a pursuit scenario). To test the behaviour of the controllers in such cases, we use an accelerating square trajectory (see Fig. 6.5), where the time taken between each successive vertex is reduced. Thus for line segment $k+1$, the time taken to complete the segment is $t_{k+1} = 0.98t_k$, with an initial time of $t_0 = 5$ s. The heading of the formation is discontinuous with a step of $\psi^d_{k+1} = \psi^d_k + \frac{\pi}{2}$ at each corner of the square, and three waypoint interpolation methods are used for the translational trajectories:

1) continuous position $\mathbf{p}^d(t)$ (Fig. 6.5a-b)

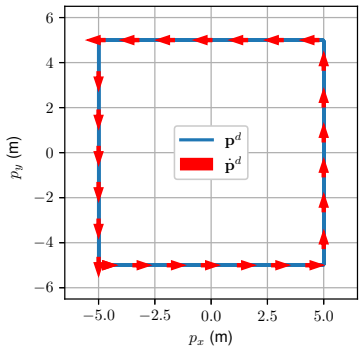
2) continuous velocity $\dot{\mathbf{p}}^d(t)$ (Fig. 6.5c-d)

3) continuous acceleration $\ddot{\mathbf{p}}^d(t)$ (Fig. 6.5e-f)

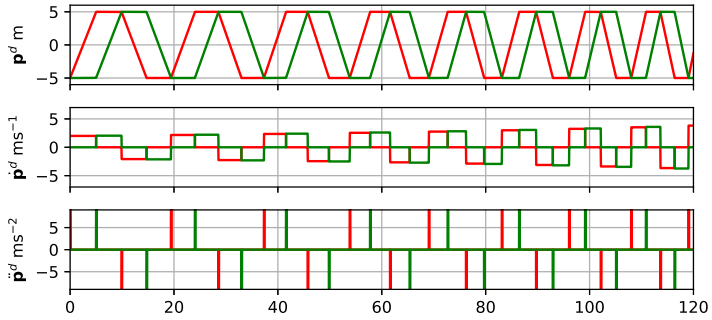
These respectively consist of linear, third order and fifth order polynomial interpolations between the two vertices over a timespan of t_k. As in the previous section, the nullspace commands set by the autopilot are set by Eq. (6.2).

The results of the square trajectories[3] are shown in Fig. 6.6 and show that both the SOVS controllers and the MPC controllers may be used, depending on the trajectory. For the continuous position trajectory the MPC controller tracks the desired position better than the SOVS controller, however this comes at the expense of a step error of approximately 0.6 in the bearing error at each heading change, showing that the SOVS controller is better for large steps in desired yaw rate. For this trajectory, the SOVS controller and MPC controller become unstable around 100 s (corresponding to velocity steps of 3.5-4.0 ms^{-1}). Interestingly, the performance of the SOVS and MPC controllers are similar for the continuous velocity

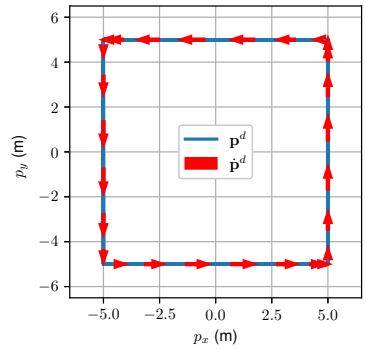
3. Videos of the MPC and SOVS square trajectory simulations may be found at https://box.ec-nantes.fr/index.php/s/fTQbXLNEqSBWwTj.



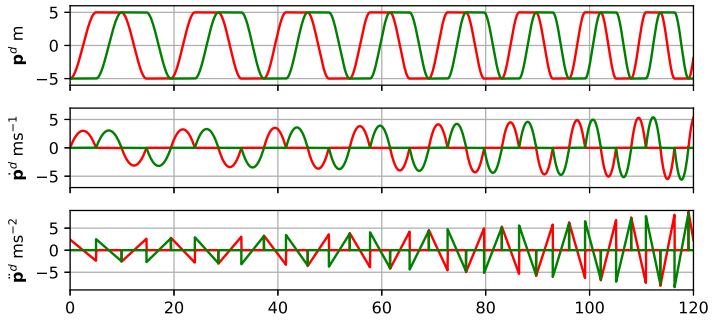
(a) 1st period of the continuous position trajectory



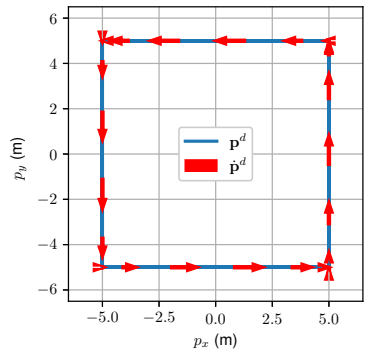
(b) Position, velocity and acceleration profile for the continuous position trajectory in the $\mathbf{x}_0$ (red) and $\mathbf{y}_0$ (green) directions



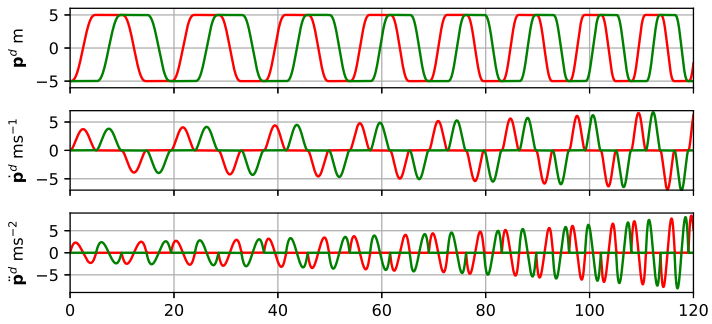
(c) 1st period of the continuous velocity trajectory



(d) Position, velocity and acceleration profile for the continuous velocity trajectory in the $\mathbf{x}_0$ (red) and $\mathbf{y}_0$ (green) directions

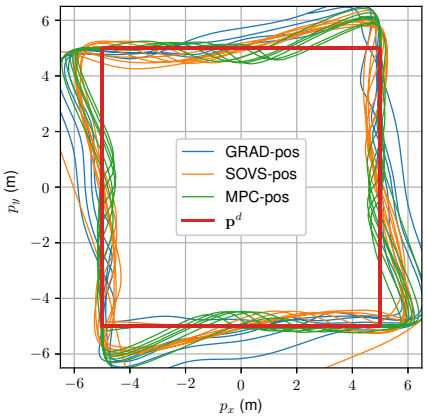


(e) 1st period of the continuous acceleration trajectory

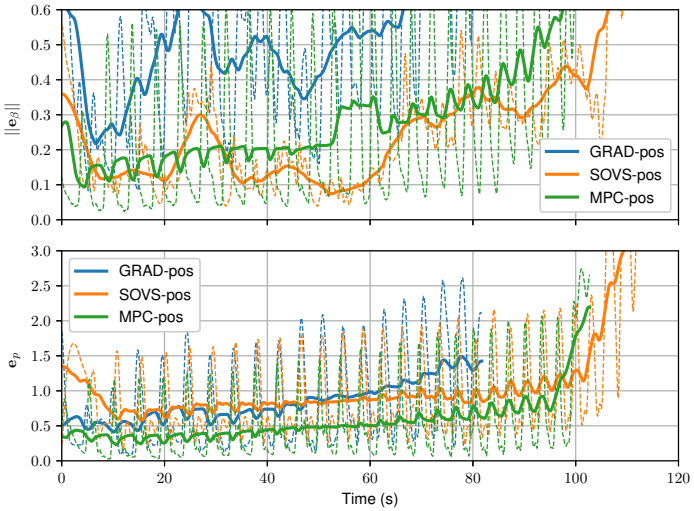


(f) Position, velocity and acceleration profile for the continuous acceleration trajectory in the $\mathbf{x}_0$ (red) and $\mathbf{y}_0$ (green) directions

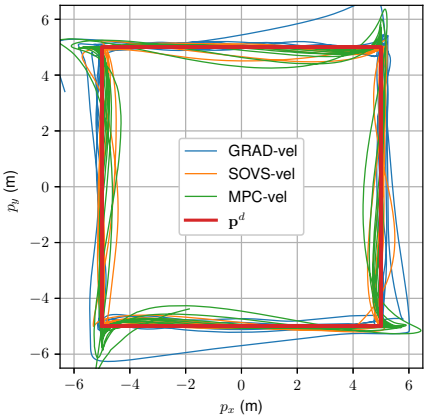
Figure 6.5 – Plots of the non-smooth. In the left hand plots, the blue line indicates the path of the trajectory and the red arrows indicate the direction of the velocity and heading of the formation. The length of each arrow is proportional to the desired velocity at that point.



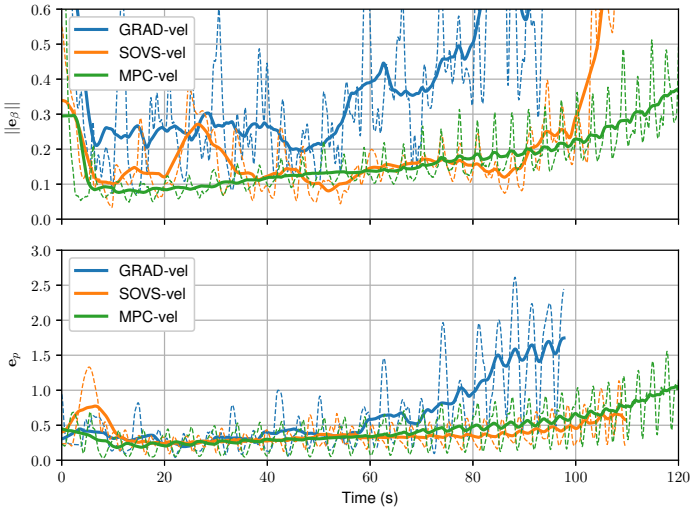
(a) Path with continuous position



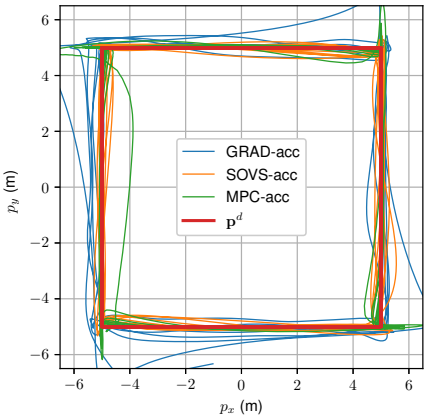
(b) Formation error (top) and tracking error (bottom)



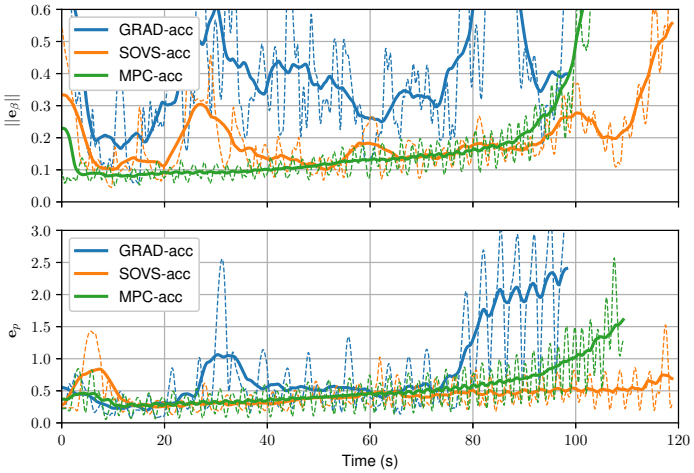
(c) Path with continuous velocity



(d) Formation error (top) and tracking error (bottom)



(e) Path with continuous acceleration



(f) Formation error (top) and tracking error (bottom)

Figure 6.6 – The tracking results for square trajectories. The filtered response is drawn with solid lines, and the raw with dashed lines.

and continuous acceleration trajectories over the first 90 s. Beyond 90 s however, the MPC controller has better stability for the continuous velocity trajectory while the SOVS controller has better stability for the continuous acceleration. Furthermore, the stability of the MPC controller is better for the continuous velocity trajectory than for the continuous acceleration. Note that the peak velocity in the continuous acceleration trajectory rises faster than in the continuous velocity trajectory which explains the earlier loss in stability for the MPC controller under continuous acceleration. We can also infer that the MPC controller may be improved by integrating acceleration information into the prediction as is done in [135], and by computing the nullspace control using acceleration as done in SOVS.

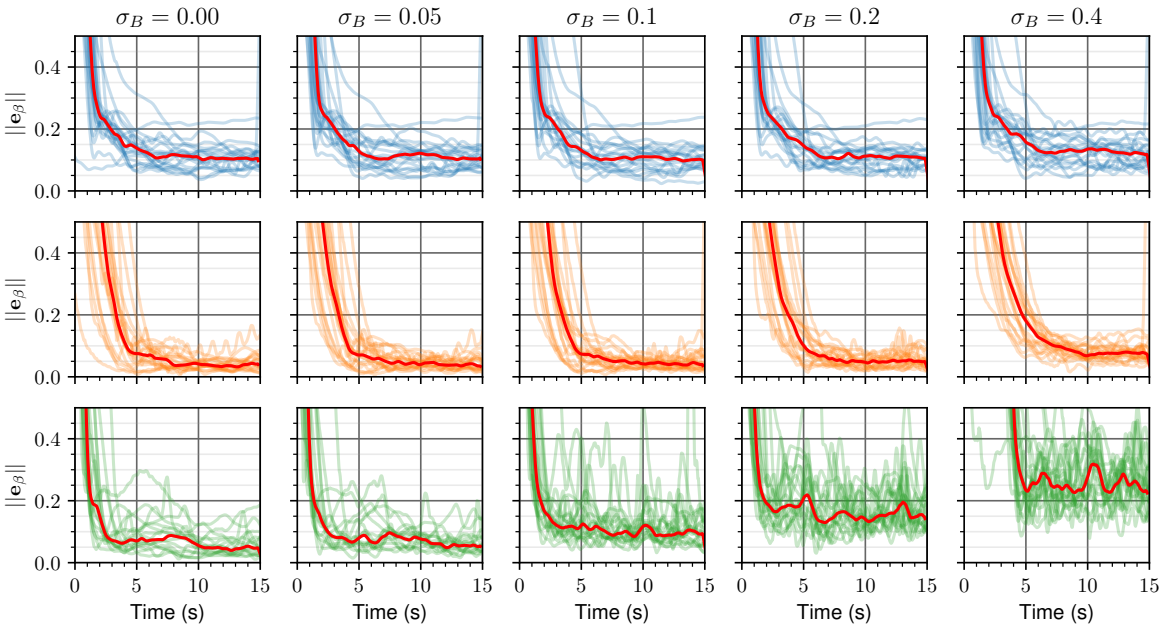
6.3 Robustness

In this section we evaluate the robustness of the formation controllers to perturbations and unmodelled effects. This is done both in static and dynamic tests to assess the stability and performance of the controllers under various uncertainties.

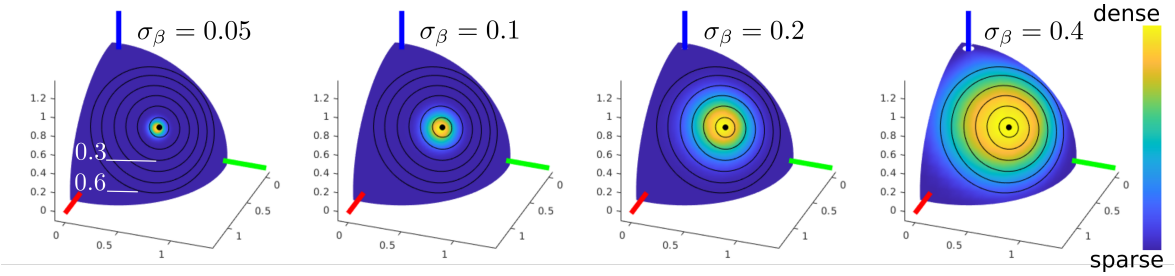
6.3.1 Robustness to Measurement Noise

Bearing measurement noise is perhaps the most obvious of the uncertainties that may occur in bearing formation control. In this section the focus is on comparing the controllers so the reader is referred to appendix B for a detailed evaluation of bearing detection methods and their associated noise, while we now focus on just the effects of a nominal gaussian noise on the controller. As explained in the appendix, there are two main types of bearing measurement uncertainty, low-frequency and high frequency noise. The former comes from mis-identification of camera distortion parameter, the camera frame to robot frame transformation, and inherent latency in the measurement pipeline, while the later comes from false or inaccurate detections and high angular dynamics of the camera. Generally the control laws are stable for bearings over the full measurement manifold, so the low-frequency noise may result in steady-state error but will be unlikely to affect the stability of the controllers. High-frequency noise however has the potential to destabilize the controllers, causing divergence from the bearing or steering tasks and potentially crashing the UAVs.

In order to evaluate the robustness of the controllers to high-frequency bearing measurement noise, a series of simulations are run with the bearings being generated from



(a) The bearing error with respect to time for a sequence of 20 random steps in desired formation for the three controllers (rigidity, SOVS and MPC corresponding to the three rows), and with varying amounts of noise.



(b) The relative density of bearing noise projected onto the unit sphere around $\mathfrak{F}_i$ for a given bearing $\boldsymbol{\beta}_{ij}$ (represented by the black dot) for various standard deviations. The concentric circles around $\boldsymbol{\beta}_{ij}$ represent bearing error magnitudes at intervals of 0.1 (see leftmost drawing).

Figure 6.7 – The effect of bearing noise on the controllers.

the relative pose of the UAVs (i.e. using MOCAP). Gaussian noise is added in an arbitrary direction orthogonal to the bearing measurement, and the standard deviation of the noise is increased for each set of experiments. The results of the simulations is shown in Fig. 6.7a, while Fig. 6.7b shows the relative density of measurements for a given nominal bearing with various standard deviation values. It is seen that the MPC controller is much more affected by noise than the other two controllers, nonetheless there was no loss in stability, only a loss in steady state tracking performance (for noise with $\sigma_\beta \leq 0.2$) and in transient performance (for very high noise with $\sigma_\beta > 0.2$). The poor performance of the MPC controller may be due to it being a more reactive controller and therefore more affected by high-frequency noise, but it could also be due to the RTI SQP optimization routine, which is only efficient if the initial guess for the control values of an optimization step is close to the optimal solution. When the standard deviation of the noise becomes large, the optimization itself has poorer convergence properties as the current and previous measurement states are significantly different leading to largely different control values. It appears from these simulations that the rigidity and SOVS controllers are little affected by bearing measurement noise of $\sigma_\beta \leq 0.2$, although in the case of the rigidity control, this is highly affected by the amount of control signal filtering.

6.3.2 Robustness to Depth Estimation

The primary reason that bearing-only control is preferred over bearing-based control is that it is generally robust to the depth of the measurement which is difficult to estimate [114]. Because of the perspective nature of bearing measurements, an error in depth estimation only scales the interaction model in the translational directions orthogonal to the bearing [192]. The depth nonetheless appears in the control laws as a part of the interaction model (Rigidity and SOVS) or prediction model (MPC) of sll the formation controllers. Some estimation of the formation scale and therefore the bearing depth is also needed to perform coordinated rotations and expansions in $\ker(\mathbf{B})$, and can generally be considered to be obtained through one of two methods:

1) An estimate of the inter-robot depth using sensor feedback such as one or more measured inter-robot distances, structure-from-motion algorithms, or operator feedback.

2) Assuming that the inter-robot distances are of constant value. It is the later method that has been used throughout this manuscript, as it is the most general.

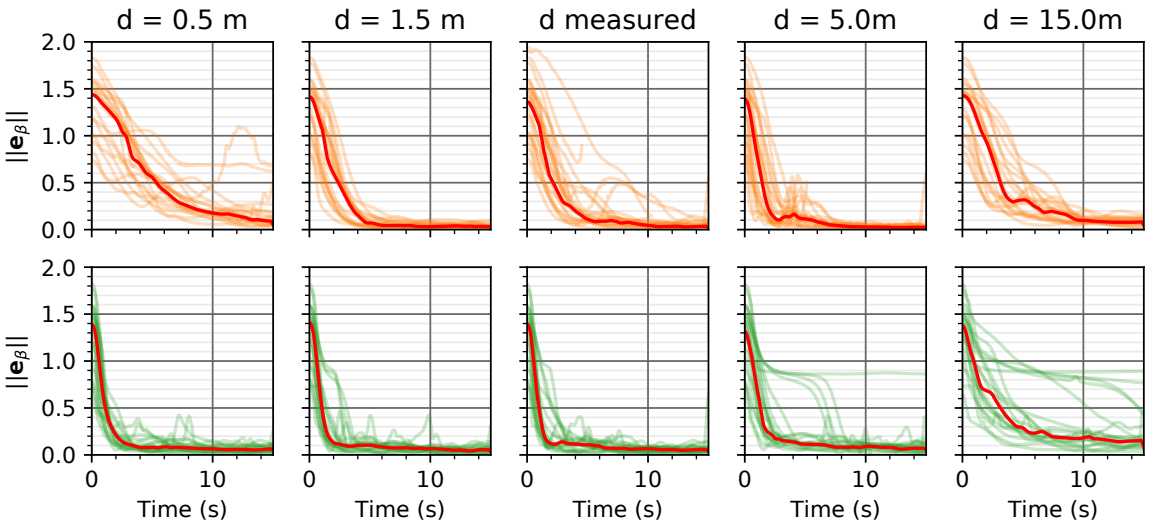


Figure 6.8 – The response to a sequence of 20 random desired formation steps for the SOVS (orange) and MPC (green) controllers, with various estimates of the inter-robot distances.

The effect of the accuracy of the depth estimates is presented in Fig. 6.8. As the distance only appears in the interaction model of the rigidity controller as a scale gain, we do not do test that controller. The SOVS and MPC controllers are tested with a sequence of 20 random desired formations at a scale automatically regulated to an average distance of 1 m to the formation COM. These tests are repeated with constant assumed distances estimates of 0.5 m, 1.5 m, 5.0 m and 15.0 m, as well as one where the distance is measured with a random gaussian noise of $\sigma_d = 0.1$ m is also provided. The MPC behaves in a very predictable manner, with fast transient convergence and low steady state error for low inter-robot distance estimates. When the inter-robot distance is overestimated by a factor of three ($d = 5$ m), some local minima begin to appear, as the prediction horizon is too short for the terminal bearing cost minimize sufficiently to compensate for the high predicted steering and control costs. When the distance is very largely overestimated ($d = 15$ m), the local minima become more difficult to avoid, and the transient convergence slows, and steady state errors become larger. The effect of the distance estimate on the SOVS controller is more complex, as the distance appears as both a scaling factor to the interaction matrix and as a scaling factor for the bearing derivative estimate (present in the derivative component of the auxiliary control law) which will result in over-damping if low and under-damping if high. The steady state error of the SOVS controller however seems to have a very low sensitivity to the distance estimate, however for large distance estimates the bearing overshoot becomes quite prominent.

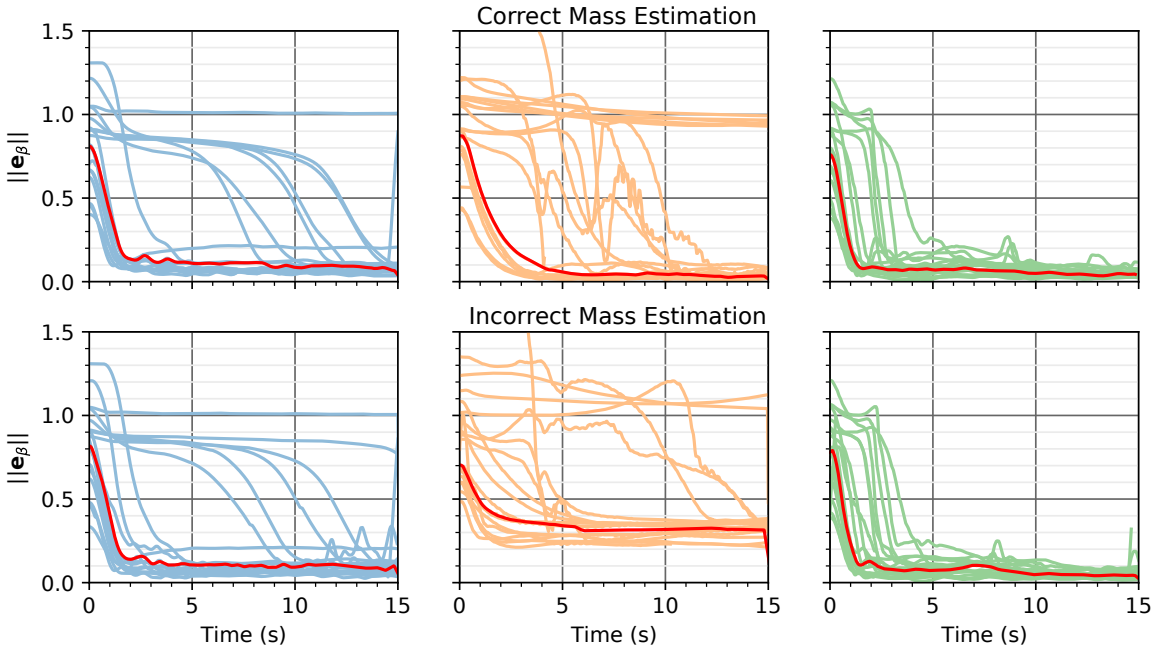


Figure 6.9 – The effect of correct (top row) and incorrect (bottom row) robot mass estimates for the rigidity (blue), SOVS (orange), and MPC (green) controllers. The median response is drawn in red.

6.3.3 Robustness to Robot Modelling Errors

A poorly estimated inter-robot depth means that a given translation motion of the camera produces an unanticipated effect on the bearing measurements. There may also however be robot modelling errors which result in the motion of the robot being different from the intended motion generated by the control input. These can include unmodelled forces such as wind buffeting that may act on the robot, but can also include problems such as an imperfect thrust identification, improperly measured mass, or inertial sensor bias. All these will result in the robot moving otherwise than desired, and may affect the tracking error or stability of the controller. In the MPC controller, this was compensated by a disturbance estimator. The SOVS controller compensates for unmodelled effects through an integral component of the auxiliary control law, and the rigidity control assumes a robust low-level controller which can track the computed velocity setpoints, in this case with a PID control.

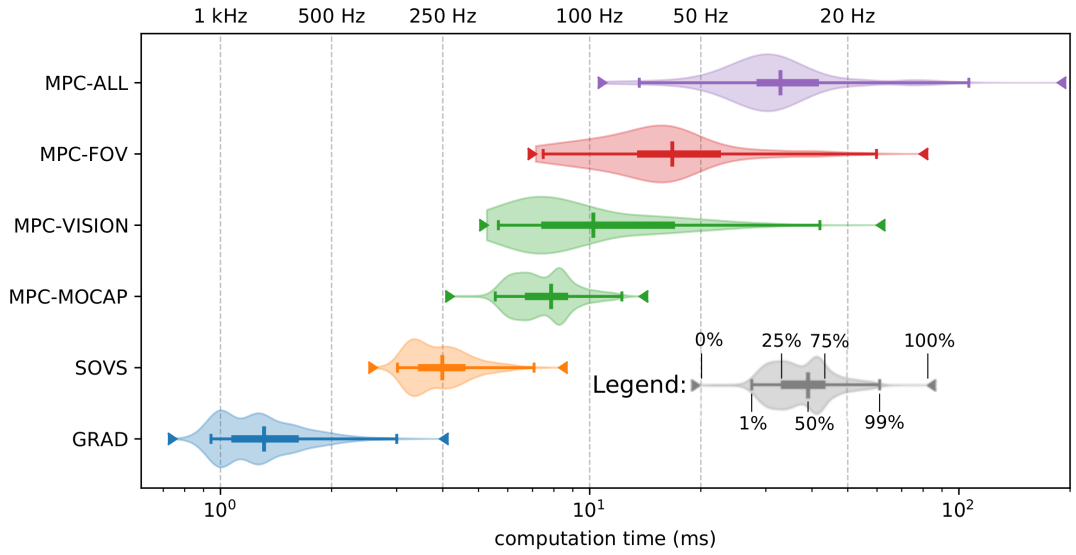
In order to evaluate the effect of an inaccurate robot model, we simulate a sequence of 20 random desired formations with each controller for two cases (note that this set of desired formations had a large set of local minima for the rigidity and SOVS controllers, unrelated to the robot model). In the first case each quadrotor knows its exact mass, and in the second case, one quadrotor has a correct estimate of its mass (1.55 kg), one overestimates its mass

by 200 g and one underestimates by 200 g. Overestimating the mass could be equivalent to an positive vertical disturbance (e.g. due to ground effect or using large-pitch propellers) and the underestimation could be considered a negative vertical disturbance (e.g. from a downdraught or loss of thrust due to damaged propellers) near hovering conditions. The results of the robustness analysis are presented in Fig. 6.9, and show that the MPC controller compensates very well for the modelling error due to the disturbance estimator. The Gradient controller was very slightly affected, some additional oscillations in bearing error due to the UAVs tracking less well the desired velocity set by the formation controller. Nonetheless the transient convergence and steady state error is negligibly affected by the inaccurate robot dynamic model. The SOVS controller on the other hand was strongly affected by the inaccurate model, with the median steady state bearing error increasing from 0.03 to 0.31. An attempt to minimize this error could be compensated by increasing the error integration saturation limit and the integral gain, however this lead to unstable convergence when the desired formation step was large. It is likely however that an external force estimator as is used by the MPC controller could greatly improve the performance of the SOVS controller in such situations.

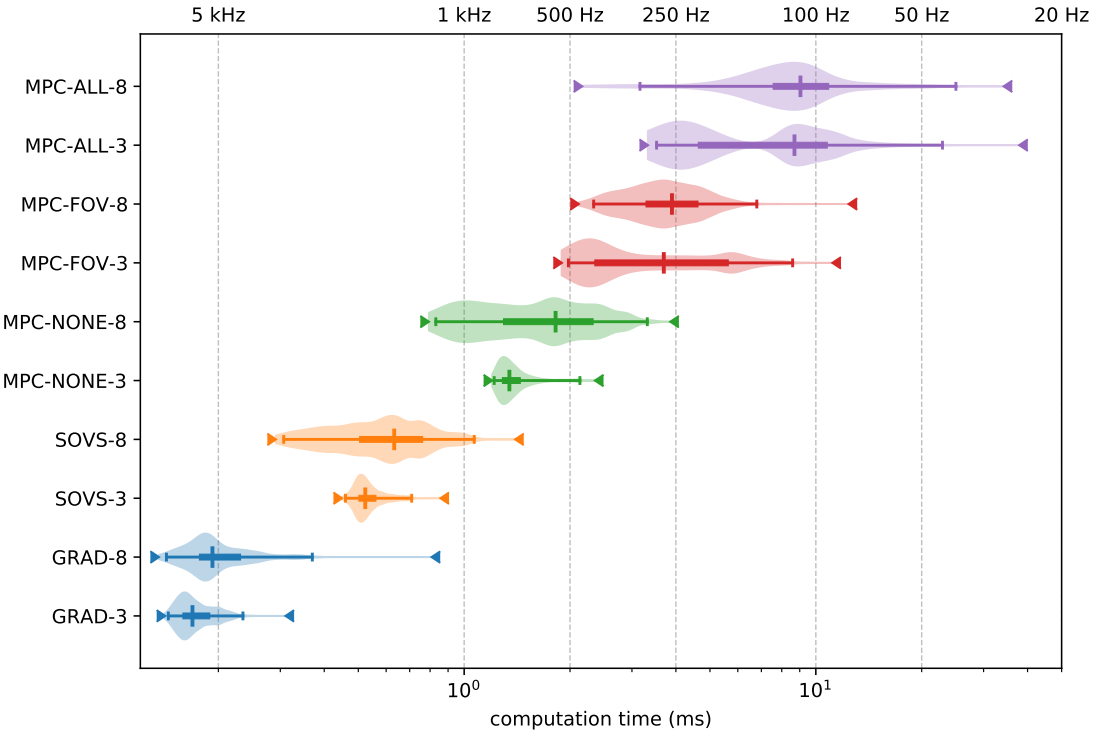
6.4 Computational Complexity

It is also important to consider the complexity of the controller, particularly with regards to the computational resources required. In this section we compare the computation time of all the controllers discussed in this work, using both the onboard Raspberry Pi computer and a groundstation laptop running SITL simulations (the hardware specifications are detailed in table 6.1). In all cases, the controllers are programmed in C++, share common sensing and communication functions, and run using ROS at a maximum frequency of 50 Hz. While an attempt was made to write efficient code in all cases, none have been optimally implemented. All analysis of computation times is therefore highly dependant on hardware, software implementation, and other parallel computational processes. The following analyse should be interpreted more as a comparative assessment of real-time feasibility rather than absolute time requirements.

The computation time in milliseconds is presented in Fig. 6.10 for various controllers. A standard box and whisker plot shows the quartiles, and the thickness of the shaded



(a) Control loop computation time on the onboard computer. All tests are performed using a three-agent complete graph formation. MPC-MOCAP has no constraints and uses motion capture to extract bearings, MPC-VISION has no constraints and uses onboard visual detection to extract bearings, MPC-FOV has FOV and altitude constraints with onboard vision, and MPC-ALL augments MPC-FOV with one spherical and one window obstacle.



(b) Control computation time on the groundstation computer. All MPC tests use MOCAP for bearing detection, and the suffix (3 or 8) indicated the number of UAVs in the formation. The thee-agent formation consists of a complete graph, while the eight-agent formation has two vertices each with one, two, three, and four directed edges leaving them.

Figure 6.10 – Distribution of calculation times for various controllers, represented on a logarithmic time scale. The quartiles are represented by a box-and-whisker diagram, and the probability density function is represented by the shaded area (see legend in fig. a)

Table 6.1 – Computer specifications for onboard and SITL control implementations

	Onboard Computer	**Ground Station**
Computer	Raspberry Pi 4	Dell Precision 7550 Laptop
CPU	Quad core Cortex-A72 1.5 GHz	Six core Intel-i7 10850H, 2.70 GHz
RAM	4 GB	32 GB

region represents the probability density function. On the onboard computer (represented in Fig. 6.10a) the computation times are tested with two measurements by each UAV[4]. For the rigidity, SOVS, and unconstrained MPC controllers, the control itself is the only significant process running on the onboard computer. For another unconstrained MPC, a FOV and altitude constrained MPC (three slack variables), and a FOV and altitude constrained MPC with obstacle avoidance (six slack variables), the onboard visual detection process runs in parallel with the controllers at a rate of 5 Hz. It can be seen that with limited computational resources, the additional demand of the visual detection algorithm reduces the performance of the MPC algorithms. Nonetheless we have shown in experiments that the MPC with FOV constraints and onboard vision can fly successfully but that when additional slack variables are included, the median frequency of 30 Hz degrades the performance of the controller. Overall, there are no problems running the rigidity or SOVS controllers, and even a less powerful computer could be used, but when using onboard vision with MPC and numerous soft constraints, it may be beneficial to use a more powerful embedded computer[5].

In the second part of the computation time experiments, the controllers are run on a laptop with the UAVs being simulated in Gazebo with SITL firmware. This is tested for a formation with three UAVs each observing the two others, and for a formation of eight UAVs, with two UAVs each measuring one, two, three, and four others. There are no visual detection processes running, but there is a significant (but equal amongst different experiments with the same number of UAVs) computational load from the simulator. Because the computer is much more powerful in this test, there are no problems with running any of the controllers (including eight parallel instances of the highly constrained MPC) in addition to the load of the simulator. For the rigidity, SOVS, and unconstrained MPC controllers, the additional edges seems to slightly increase the median computation times and stretch out the range of

4. The computation time of all UAVs are grouped together for each experiment

5. Note that this conclusion is only with respect to control computation, and ignores the benefits of an upgraded computer (particularly with a GPU) on the onboard visual detection performance. A good candidate which would be sufficient for all control and visual processes would be a Jetson Xavier NX or similar for UAVs that are sufficiently powerful.

computation times, although the latter may be due to the increased additional load of the simulation and due to the number of loops in the sensing and communication components of the controller which have the potential to be significantly improved. For both of the three UAV constrained MPC controllers, two grouping can be seen, likely corresponding to the activation or lack thereof of the soft constraints (section 5.6) which will directly affect the runtime of the RTI SQP algorithm. When there are eight UAVs for the constrained MPC algorithms, the computation times are generally slower with a normal distribution. The lack of multiple groupings due to soft constraint violations is likely due to the distribution of strained computational resources by the operating system, as a result of the computation load.

6.5 Conclusions

In this chapter, a comparison between the different controllers is performed to assess their performance and robustness in static and dynamic operating conditions, as well as the required computational resources for implementation. The formation controllers were tested extensively in simulations which have similar characteristics (within the tested operating conditions) as real UAVs. While there is no best controller per se, we have attempted to provide a comparison between the two developed controllers (SOVS and MPC) and the baseline controller (rigidity) to establish the benefits of each, which are summarized hereafter.

In general, the MPC controller performs the best under static and smooth dynamic conditions, supposing that reasonable depth estimations and bearing detection precision can be ensured[6]. The MPC controller requires significant computational resources however (particularly when soft constraints are included), and care should be taken to validate computation times, particularly if the onboard computer does not have a real-time OS. The SOVS controller also had good convergence in static and dynamic situations, however unlike the other controllers, its convergence rates are less strongly related to the scale of the formation. When abrupt changes in formation heading are desired, the SOVS controller maintains the formation geometry better than the MPC controller, at the expense of losing some manoeuvring performance. It also is more robust than the MPC controller to uncertain

6. See the respective sections of this chapter

depths and noisy bearing measurements, but only when the quality of these become very poor. The integrator in the auxiliary control law is insufficient to compensate for large perturbations, thus a disturbance estimator should be included in future implementations, as is done with the MPC controller. The gradient controller performs well in static conditions for small formations, however the convergence becomes poorer compared to the others as the scale increases, or if significant manoeuvring of the formation is desired. Nonetheless it has the advantage of not requiring any scale estimate, being quite robust to noise and perturbations, and is computationally trivial.

PART III

Singularities in Bearing Formations

Abstract

In the preceding sections we have shown various dynamic formation control strategies, however they have all been predicated on the assumption of bearing rigidity. Chapter 7 begins by demonstrating the effect of a loss in rigidity on a formation controller, highlighting the need for further study. While the combinatorial rigidity of any given formation can be easily assessed, singular embeddings where rigidity is lost are difficult to identify a priori. Starting with simple examples, we show how these configurations have been previously identified, but that existing analyses of singular configurations are limited to very small formations.

The primary contribution of this part comes in chapter 8, in which we develop a novel method of characterizing singularities, identifying them, and designing formations considering them. A screw theory-based approach is used to express the constraints of simple groupings of edges and vertices in the formation graph, and contractions of constraints can then be used to get a more thorough understanding of the singularities of more complex formations.

List of Chapters

CHAPTER 7

GRAPH RIGIDITY AND KINEMATIC SINGULARITIES

SINGULARITIES have been a highly studied topic in mechanics for many years. In the case of multi-robot systems however, there has been comparatively little study on how to identify singular configurations, and their effects on the system. In this chapter, we present a state-of-the-art on singularities in bearing formation control, showing current examples of how they are treated in the multi-agent community. We begin by a practical example of the effect of a known singular configuration on a simple formation. It is then shown how the hidden robot concept has been adapted to apply mechanical singularity analysis tools to the study of singularities in visual control, and more recently in formation control. The chapter ends by showing the drawbacks of the current method of bearing formation singularity analysis, which scales poorly beyond formations of three or four agents, and requires a strong basis in kinematic analysis to accomplish.

7.1 Rigidity and Singularities in Formation Control

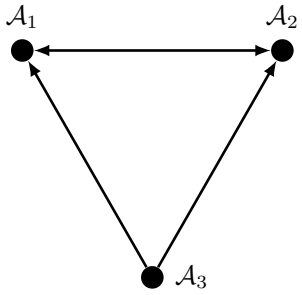
The fundamental interest of bearing formation control is the ability to steer a group of robots as if they were a single rigid body. We recall from chapter 3 that if a bearing formation is infinitesimally rigid, the embedding of its vertices are uniquely defined by the measurements of its edges, up to an arbitrary trivial motion consisting of a displacement, yaw rotation, and scale. Rigidity is a property of the formation graph, and many works have studied it from a combinatorial perspective. Tools such as Henneberg construction can be used to design bearing graphs that are guaranteed to be generically rigid [111], [193], [194]. Other works such as [195] identify rigid subformations of non-rigid graphs, and suggest how to enforce rigidity by adding new measurement. These works however deal only with generic embeddings of formation graphs on given manifolds, whereas it is well known that rigidity is dependant on the exact embedding, as proven in [196] which states:

> "A graph in $\mathbb{R}^n$ is either rigid for *almost all* locations of its vertices or flexible for *almost all* locations of its vertices." [*emphasise added*]

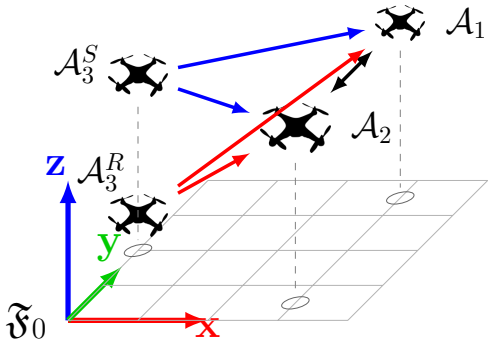
This is interesting, as it implies that graphs developed to be rigid from a combinatorial perspective may nonetheless have a set of embeddings at which they become flexible (and as is the case in Fig. 7.1). It is these sets of embedding that we refer to as singularities, and the identification of the geometric conditions leading to them is one of the objectives of this dissertation.

While rigidity has been identified as a necessary condition to guarantee the bearing error convergence of gradient-based control laws [121], this doesn't seem to have significant importance in practice[1] for the minimization of bearing error or in the manoeuvring of the formation. It remains a critical aspect of all bearing formation controllers however, as is demonstrated in Fig. 7.1. Two simulations[2] are performed with a formation in a rigid and a known flexible (singular) configuration, both using the MPC formation control method. It can be seen that despite the same graph structure, the rigid formation maintains the desired formation shape (an equilateral triangle), while the flexible formation changes shape (with two UAVs eventually colliding) despite maintaining the same bearing error performance (RMS bearing errors of 0.035 and 0.029 for the rigid and singular flights respectively).

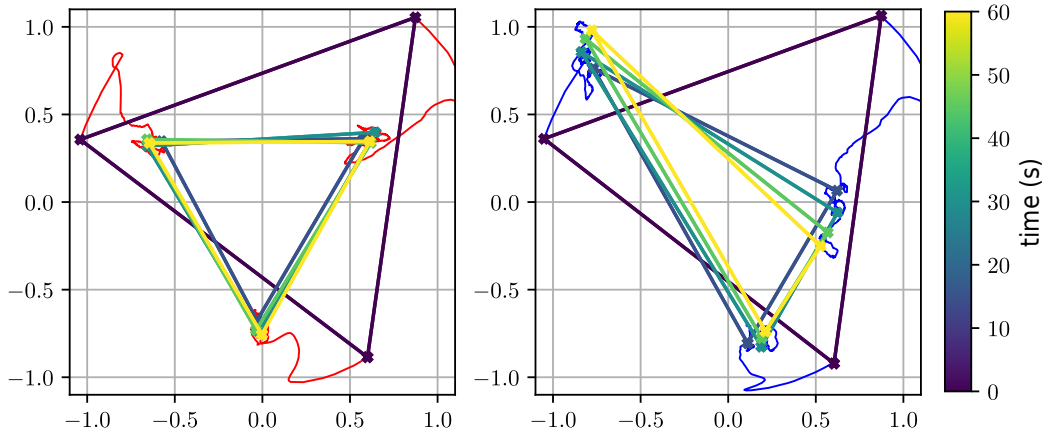
1. Simulations of rigidity-based control with single-integrator robots showed local minima at singularities, but these disappear for robots with real dynamics. While we do not deal extensively with bearing-based localisation in this dissertation, these estimators are likely more affected than the control as they make use of single-integrator observers.
2. Videos of similar simulations may be found at https://box.ec-nantes.fr/index.php/s/AY3EoLqZLRmHCFB.



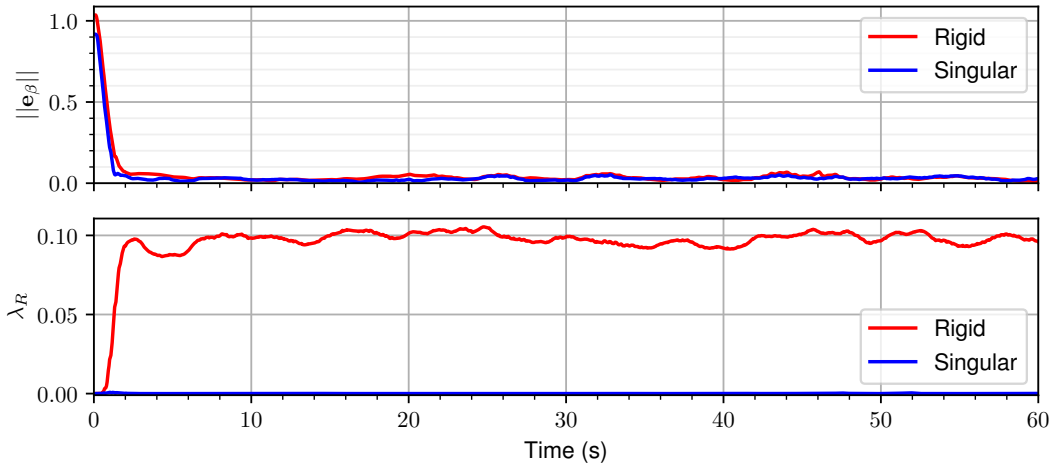
(a) The desired formation framework projected on the x-y plane. Both formations in (b) appear as this from a top view.



(b) The rigid (red) and singular (blue) desired formation frameworks. The black arrow is the same for both flights.



(c) The cartesian position of the UAVs projected on the x-y plane for the rigid (left) and singular (right) desired embeddings as a function of time (color). The red and blue traces represent the paths of the UAVs for the rigid and singular simulations. The bottom corner of the formations correspond to the positions of $\mathcal{A}_3$.



(d) The evolution of the bearing error (top) and the rigidity eigenvalue λ_R (bottom) for the rigid and singular flights. Between 5-60 s, the RMS bearing errors are 0.035 and 0.029 for the rigid and singular flights respectively

Figure 7.1 – Simulations (using Gazebo and PX4 SITL, with an MPC bearing formation controller) demonstrating the effect of a singularity in formation rigidity. The formations are subjected to a nullspace rotation of 0.25 rad/s in order to excite the system, and which is removed from the cartesian plots to better compare the evolution of the formation shape.

We believe that the study and understanding of the causal conditions and the effects of singularities is of importance to the field of multi-robot control by:

- Allowing the design of controllers that can guarantee rigidity, but can still pass through singular conditions (i.e. momentarily lose rigidity) as existing rigidity maintenance controllers [120], [197], [198] are based on potential functions that naturally inhibit the crossing of singular locii. Furthermore these types of controller require complicated decentralized estimates of global properties such as the rigidity eigenvalue λ_R (see section 3.4.3) [120] and a geometric interpretation may lead to simpler methods.

- The design of formations to avoid or accept certain singularities based on their effects. As shown later, not all singularities have the same effects, and an understanding of this may lead to formations that are either permitted because the singularities are unimportant, or disallowed because of singularities which are dangerous or excessively limiting.

- Contribute to the general scientific knowledge in the field of graphs, frameworks, and rigidity, which has been studied extensively for the past 50 years and yet has no systematic means of understanding the nature of bearing framework singularities from a geometric perspective.

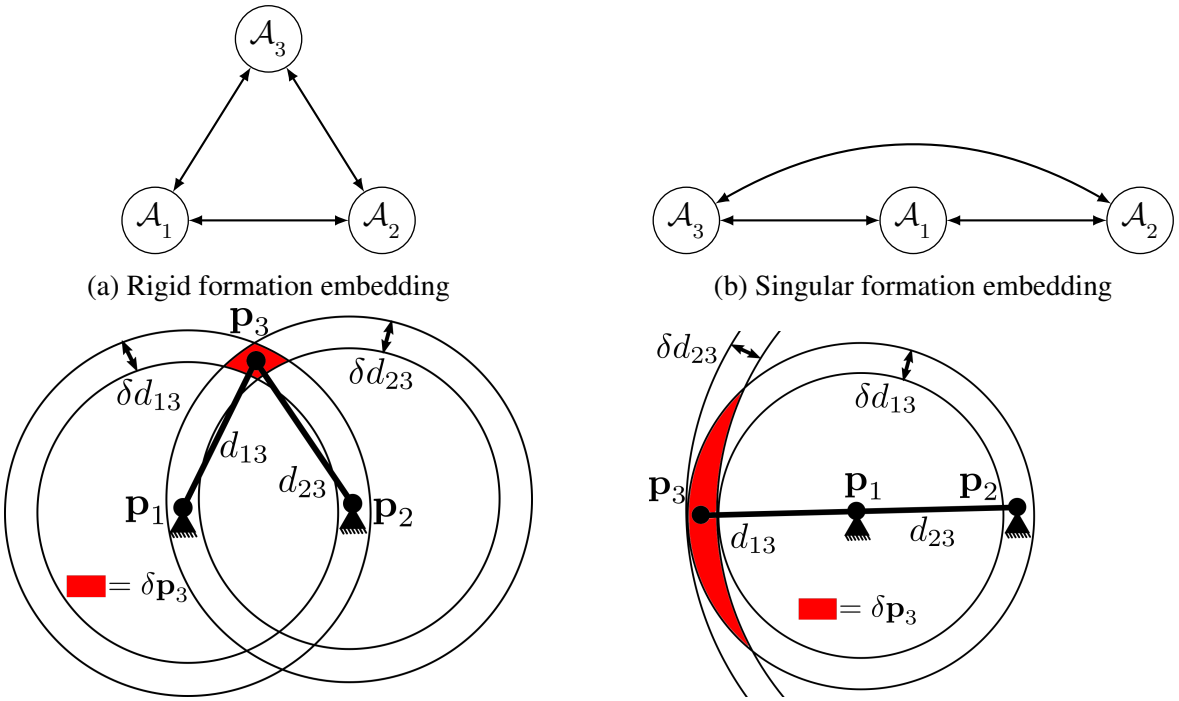
Having presented the importance of rigidity in bearing formation controllers, we now take a step back from multi-robot systems and present a physical interpretation of rigidity that will help the development of the analysis in chapter 8.

7.2 Simple Examples of Rigidity Singularities

The easiest way of presenting the physical nature of rigidity comes from truss structures, used extensively in civil and mechanical engineering. Each truss member constraints the distance between its two endpoint, and as such naturally compares to distance-based formation control, discussed briefly hereafter.

7.2.1 Distance Formation Control

We begin by considering the case of three coplanar robots maintaining fixed relative inter-agent distances. If we consider that the agents are arranged in a generic triangular



(a) Rigid formation embedding

(b) Singular formation embedding

(c) Deflection uncertainty far from singularity

(d) Deflection uncertainty at singularity

Figure 7.2 – Rigid and singular configurations of a 3-robot distance-based formation, with their corresponding pin-joint truss structure and region of uncertainty. For simplicity, agent $\mathcal{A}_1$ and $\mathcal{A}_2$ are fixed in position.

embedding on $\mathbb{R}^2$ as shown in Fig. 7.2(a-b), the three fixed-distance edges will necessarily form a unique triangle (with a trivial motion of two translations and a rotation). This can be interpreted as a pin-joint truss as the three vertices are constrained by their relative position but not by their relative orientations, and the trivial motions may be eliminated by fixing two vertices in position, as done in Fig. 7.2c-d forming a static truss. Because no controller will be able to perfectly track a reference, we will assume that the distances are regulated within a margin $d_{ij} \pm \delta d_{ij}$ (or the truss equivalent of slight elasticity in the links). Considering δd_{13} and δd_{23} as an infinitesimally small time-varying value and assuming that $\mathbf{p}_1$ and $\mathbf{p}_2$ are fixed, the distance rigidity equation can be expressed as

$$\begin{bmatrix} \dot{d}_{13} \\ \dot{d}_{23} \end{bmatrix} = \mathbf{D} \begin{bmatrix} \dot{p}_{3x} \\ \dot{p}_{3y} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \quad \text{where } \mathbf{D} = \begin{bmatrix} \mathbf{u}_{13}^T \\ \mathbf{u}_{23}^T \end{bmatrix} \tag{7.1}$$

where $\mathbf{D}$ is the distance rigidity matrix [111] and $\mathbf{u}_{ij}$ is the unit vector in the direction $\mathbf{p}_j - \mathbf{p}_i$. The solution in a general case where $\text{rank}(\mathbf{D}) = 2$ is that $\dot{p}_{3x} = \dot{p}_{3y} = 0$, and thus the state of the system is fully defined. The singularity of this formation (or of this truss) is well known, and occurs when all agents are co-linear which can be verified by setting $\mathbf{u}_{13} = \mathbf{u}_{23}$. This

reduces Eq. (7.1) an equation with an infinite number of solutions, where any velocity $\dot{\mathbf{p}}_3$ may be admissible so long as it lies within the kernel of $\mathbf{D}$ and thus has the solution

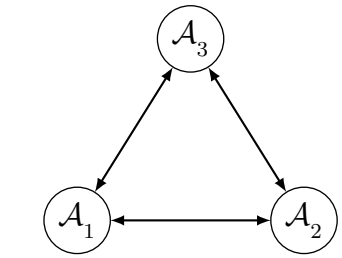
$$\dot{\mathbf{p}}_3 \in \ker(\mathbf{D}) = \gamma \begin{bmatrix} -u_{12,y} \\ u_{12,x} \end{bmatrix} \; \forall \gamma \in]-\infty, \infty[\tag{7.2}$$

In the case of a truss, a singular condition would correspond to a deflection due to small applied loads, as even for very stiff links where $\delta d_{ij} \to 0$, the positional uncertainty $\delta \mathbf{p}_3$ does not converge to a point, but rather an infinitesimal line segment. More interesting for us, in the case of a multi-robot formation any instantaneous velocity of $\mathcal{A}_3$ orthogonal to the line of both distance measurements produces no change in the measured values.

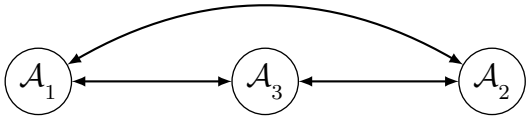
Moving back from the static truss to the formation of three mobile robots, we can see that the problem becomes slightly more complex. As the two static pylons of the truss are now mobile, each agent is mobile with its own distance measurements and motion control. We will not go into detail on rigidity control of distance-based formations as the objective of this section was to explain the link between multi-robot formation control and mechanical rigidity. We now move on to the physical interpretation of bearing formation control, and show with a similar example how the rbearing igidity problem is formulated.

7.2.2 Bearing Formation Control

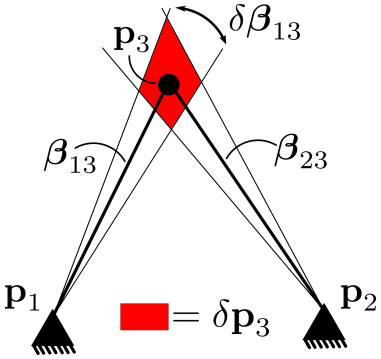
While the truss is an intuitive physical analogy for distance-based control, the physical mechanism for bearings is more complicated and will be presented in the following section. Bearings however have had centuries of use in maritime navigation, and thus we will present bearing singularities as a planar localisation problem. As with the distance example, we begin by fixing $\mathcal{A}_1$ and $\mathcal{A}_2$ to restrict the trivial motion of the formations, and thus $\mathbf{p}_1$ and $\mathbf{p}_2$ become static points. If $\mathcal{A}_3$ wishes to localize itself with respect to the other agents, it can measure its bearing with respect to $\mathcal{A}_1$ and $\mathcal{A}_2$, and its position $\mathbf{p}_3$ will lie at the intersection point of $\boldsymbol{\beta}_{13}$ and $\boldsymbol{\beta}_{23}$. In a general configuration as shown in Fig. 7.3c, a point in space can be defined by two bearings. If we consider that the bearing measurement $\boldsymbol{\beta}_{ij}$ has some uncertainty $\delta\boldsymbol{\beta}_{ij}$, then there will be some uncertainty $\delta\mathbf{p}_3$ in the localisation of $\mathcal{A}_3$. Assuming the bearing uncertainty to be of infinitesimally small value, the localization error of $\mathcal{A}_3$ can



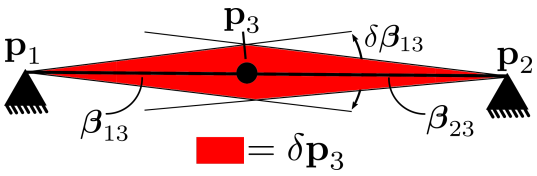
(a) Rigid formation embedding



(b) Singular formation embedding



(c) Positional uncertainty far from singularity



(d) Positional uncertainty near singularity

Figure 7.3 – Rigid and singular configurations of a 3-robot bearing-based formation, with their corresponding constraints and region of uncertainty. For simplicity, agent $\mathcal{A}_1$ and $\mathcal{A}_2$ are fixed in position.

be expressed as

$$\begin{bmatrix}\delta\boldsymbol{\beta}_{13}\\ \delta\boldsymbol{\beta}_{23}\end{bmatrix} = \mathbf{B}\begin{bmatrix}\delta p_{3x}\\ \delta p_{3y}\end{bmatrix} = \begin{bmatrix}0\\0\end{bmatrix} \quad \text{where } \mathbf{B} = \begin{bmatrix}\boldsymbol{\beta}_{13}^{\perp\,T}/d_{13}\\ \boldsymbol{\beta}_{23}^{\perp\,T}/d_{23}\end{bmatrix} \tag{7.3}$$

where B is the planar bearing rigidity matrix and $\boldsymbol{\beta}_{ij}^{\perp}$ is the unit vector perpendicular to $\boldsymbol{\beta}_{ij}$. In a general configuration this has the trivial solution of $\delta\mathbf{p}_3 \to 0$ as $\delta\boldsymbol{\beta}_{i3} \to 0$, showing that as the bearings measurements become perfectly accurate, so does the localization. In the case of the well known singularity in Fig. 7.3d where $\boldsymbol{\beta}_{13}$ and $\boldsymbol{\beta}_{23}$ become parallel, it is clear that $\text{rank}(\mathbf{B}) = 1$ and that $\ker(\mathbf{B}) = \boldsymbol{\beta}_{12}$. The position $\mathbf{p}_3$ of $\mathcal{A}_3$ can be found as

$$\mathbf{p}_3 = \mathbf{p}_1 + \gamma\boldsymbol{\beta}_{12}\ \forall\,\gamma\ \in]0, d_{12}[\ \text{if}\ \boldsymbol{\beta}_{13} = -\boldsymbol{\beta}_{23} \tag{7.4a}$$

$$\mathbf{p}_3 = \mathbf{p}_1 + \gamma\boldsymbol{\beta}_{12}\ \forall\,\gamma\ \in]0, \infty[\ \text{if}\ \boldsymbol{\beta}_{13} = \boldsymbol{\beta}_{23} \tag{7.4b}$$

and therefore even as the bearing becomes infinitely precise the positional uncertainty $\delta\mathbf{p}_3$ does not go to zero, instead becoming a line segment joining $\mathbf{p}_1$ and $\mathbf{p}_2$. The time derivative of this formulation equivalently shows that in the singular configuration, any velocity along the line segment will not alter the formation in an observable manner.

This example shows a very important distinction between the singularities in bearing and distance problems. While distance singularities allow for an infinitesimal motion changing the shape of the framework, the resulting singular motion moves the framework out of a singularity and into rigid equilibrium condition. For bearing formations however, the singular condition permits unobservable motions along a non-infinitesimal locus on the vertex manifold. In the example in Fig. 7.3d, one can see that regardless of how precise the bearing measurements become, $\mathcal{A}_3$ may collide with $\mathcal{A}_1$ or $\mathcal{A}_2$ without any means of observing its motion and thus preventing a collision.

The presented examples of singularities have hitherto only been developed considering the motion of one agent on the $\mathbb{R}^2$ manifold while the others remain fixed, greatly simplifying the problem. If one were to rigorously analyse the singularities of the three-agent bearing formation on $\mathbb{R}^3 \times \mathbb{S}^1$ manifold using linear algebra, one must consider the full dynamic system

$$\dot{\boldsymbol{\beta}} = \mathbf{B}\dot{\mathbf{q}} \tag{7.5}$$

where the measurement vector is $\boldsymbol{\beta} = [\boldsymbol{\beta}_{12}^T \dots \boldsymbol{\beta}_{23}^T \dots \boldsymbol{\beta}_{32}^T]^T \in \mathbf{R}^{18}$, the embedding vector is $\mathbf{q} = [\mathbf{p}_1^T \ \psi_1 \dots \mathbf{p}_3^T \ \psi_3]^T \in \mathbb{R}^{12}$, and the bearing rigidity matrix is $\mathbf{B} = \frac{\partial \boldsymbol{\beta}}{\partial \mathbf{q}} \in \mathbb{R}^{18\times 12}$. It can be shown for any rigid formation on the $\mathbb{R}^3 \times \mathbb{S}^1$ manifold that $\text{rank}(\mathbf{B}) = 4|\boldsymbol{\mathcal{V}}| - 5$ [118] (i.e. the embedding of each vertex minus the trivial motions of the formation), and thus within this 18×12 matrix, to understand the singularities we must find all conditions leading to $\text{rank}\,(\mathbf{B}) < 7$. To do so analytically is difficult, and becomes even more challenging as the number of UAVs increases. One can of course fix the trivial motions, and perform a 7-dimensional[3] discrete grid search evaluating the rigidity eigenvalue to determine where the formation is singular, however because of the dimensionality, this method cannot scale beyond 3-4 UAVs and does not contribute to a geometric understanding of the phenomenon of singularities. To systematically analyse the singularities of bearing formations we will therefore borrow tools from the field of mechanism kinematics, which are described in the following section.

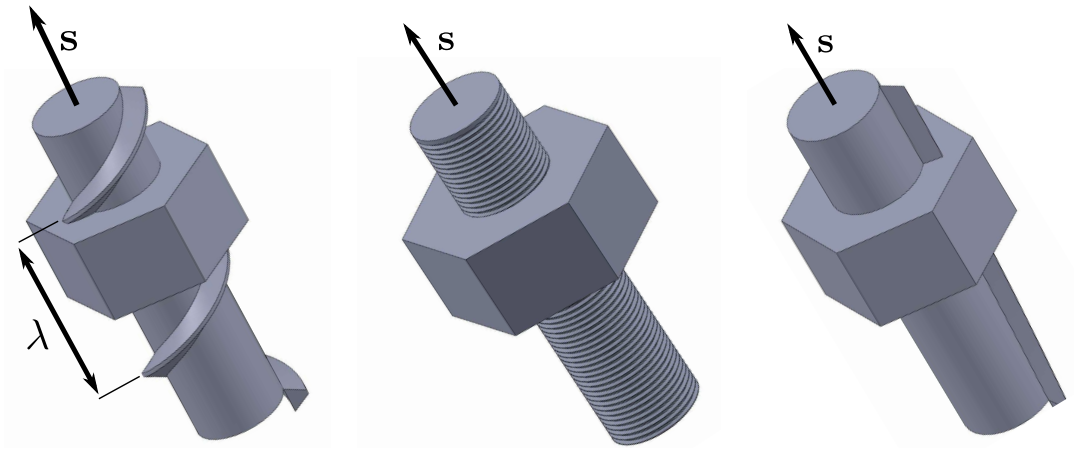
3. The rigidity is invariant to the yaw of the UAVs and the second UAV may be a fixed distance from the first fixed-position UAV so as to restrict the scale component of the trivial motion. This bring the actual dimensionality of a complete workspace singularity search problem to $3(|\boldsymbol{\mathcal{V}}| - 2) + 2$ which is still unmanageable beyond 3-4 UAVs

7.3 On the Identification of Mechanical Singularities

The first mention of kinematic singularities in the context of mechanism kinematics that I have been able to find dates from 1965, where the author analyses statically determinate mechanisms to identify configurations for which an external load may cause the mechanism to experiences high internal forces [199]. Several years later, singularities are again studied by [200], in which the kinematics of a Stirling cycle engine are analysed in less than one dollar of computer time. Through the 1970s, singularities were occasionally mentioned as conditions where a matrix looses rank in papers developing multi-body kinematic and dynamic analysis, but it is not until the 1980s that a formal theory of kinematic singularities began to be developed. The beginning of the 1990s saw a number of influential publications with mechanical singularities as the principal subject, either attempting to characterize them or to reduce their impact on robot control [201]–[204]. Since then, the development of geometric, algebraic, and numerical methods to assist in the understanding of kinematic singularities has been an active field of study for kinematicians and roboticists and has provided a wealth of tools. Nonetheless, there is still no universal method for studying singularities of complex multi-body systems. In this thesis we select screw theory as a means of studying mechanism singularities, and eventual formation framework singularities. In certain cases outlined throughout the following chapter this limits the thoroughness of the analysis, however the use of more complex mathematical tools such as Grassman geometry, Assur graphs, and Gröbner bases are left as a possible extensions.

7.3.1 The Notation of Screw Theory

While there are many methods of finding kinematic singularities, we will use screw theory in this manuscript, as it is a geometrically intuitive (and perhaps the simplest) method. In some applications, other tools may be more appropriate, however exploring all these tools risks departing far from the core of this thesis which is the control of multi-robot formations. The applicability of other specialized tools is therefore noted when it could be of added value, but not further explored. While this section develops a sufficient base of screw theory to support its use later on, it remains a tool that is little used in the multi-robot community, and many readers of this manuscript may be unfamiliar with it. An example demonstrating the analysis of singularities for a simple mechanical structure by screw theory is thus provided



(a) Finite pitch screw (b) Zero pitch screw (c) Infinite pitch screw

Figure 7.4 – A representation of the principles of screw theory

in appendix C to assist with an understanding of the methodology applied, and the reader is also referred to [205] for more practical details.

Screw theory is a mathematical framework for expressing pairs of vectors as a single algebraic feature: a screw. Developed in 1876 by Sir Robert Ball [206], screw theory represents forces and motions as sets of Plucker coordinates, and has applications in rigid-body dynamics and mechanism design [205]. The general representation behind screw theory is that the simplest representation of a motion is the rectilinear motion of a translation along an axis and a coupled rotation about the axis, as is the case with physical screws (represented in Fig. 7.4). The general formation of such a screw is

$$\$ = \begin{bmatrix} \mathbf{s} \\ \mathbf{s} \times \mathbf{p} + \lambda \mathbf{s} \end{bmatrix} \tag{7.6}$$

where $\mathbf{s} \in \mathbb{R}^3$ is a vector representing the axis of the screw, λ is a scalar value called the pitch and $\mathbf{p} \in \mathbb{R}^3$ is the position of a point of application in the frame of representation of the screw. As we use the screw to detail infinitesimal motion, the screw has a unit length. In our application, we generally deal with two special types of screws: those having a pitch of either zero or infinity. A zero pitch screw (Fig. 7.4b) may be obtained by setting $\lambda = 0$, resulting in

$$\$^0 = \begin{bmatrix} \mathbf{s} \\ \mathbf{s} \times \mathbf{p} \end{bmatrix} \tag{7.7}$$

and represents a purely rotational action, while the infinite pitch screw (Fig. 7.4c) resulting

from setting $\lambda = \infty$ is defined as

$$\$^{\infty} = \begin{bmatrix} \mathbf{0} \\ \mathbf{s} \end{bmatrix} \tag{7.8}$$

resulting in a purely translation action.

These two formulations present the mathematical framework required to apply screw theory, however it is common to separate screws into groups defined by their physical meaning. To this purpose, we present twists (screws representing infinitesimal motion) and wrenches (screws representing infinitesimal efforts) using different notations despite both being screws as defined previously.

7.3.2 Twists and Wrenches

This screw notation is used to represent velocity twists $\mathbb{v}$ (consisting of angular $\boldsymbol{\omega}$ and translational $\mathbf{v}$ velocities) and wrenches $\mathbb{w}$ (consisting of forces $\mathbf{f}$ and moments $\mathbf{m}$)

$$\mathbb{v} = \begin{bmatrix} \boldsymbol{\omega} \\ \mathbf{v} \end{bmatrix} \qquad \mathbb{w} = \begin{bmatrix} \mathbf{f} \\ \mathbf{m} \end{bmatrix} \tag{7.9}$$

A twist is a screw representing an infinitesimal motion, and like the general unit screw, we consider both the zero and infinite pitch varieties. By considering Fig. 7.4b, and imagining that λ becomes infinitesimally small, we can see that the nut will simply rotate around the screw with no translational motion. A zero-pitch twist $\mathbb{v}^{\mathrm{r}}(\mathbf{s}, \mathbf{p})$ represents a pure infinitesimal rotation with velocity ω around an axis $\mathbf{s}$ at point $\mathbf{p}$

$$\mathbb{v}^{\mathrm{r}} = \begin{bmatrix} \omega\mathbf{s} \\ \omega\mathbf{s} \times \mathbf{p} \end{bmatrix} \tag{7.10}$$

Likewise as can be seen from Fig. 7.4c, an infinite-pitch twist $\mathbb{v}^{\mathrm{t}}(\mathbf{s})$ represents a pure infinitesimal translation with velocity v along axis $\mathbf{s}$

$$\mathbb{v}^{\mathrm{t}} = \begin{bmatrix} \mathbf{0} \\ v\mathbf{s} \end{bmatrix} \tag{7.11}$$

Two wrench types can be represented similar to the twists. A zero-pitch wrench $\mathbb{w}^{\mathrm{f}}(\mathbf{s}, \mathbf{p})$ represents a pure force f along axis $\mathbf{s}$ acting at position $\mathbf{p}$ with respect to some frame of

reference, and an infinite-pitch wrench $\mathbb{w}^{\mathrm{m}}(\mathbf{s})$ represents a pure moment m around $\mathbf{s}$ [205]

$$\mathbb{w}^{\mathrm{f}} = \begin{bmatrix} f\mathbf{s} \\ f\mathbf{s} \times \mathbf{p} \end{bmatrix} \qquad \mathbb{w}^{\mathrm{m}} = \begin{bmatrix} \mathbf{0} \\ m\mathbf{s} \end{bmatrix} \tag{7.12}$$

As with twists the effect of wrenches can be visualized in Fig. 7.4, and recognizing that unlike a twist which represents an infinitesimal motion, wrenches are infinitesimal efforts that impede a motion. We can therefore consider that in the case of the zero-pitch screw in Fig. 7.4b, the wrench would be a pure force along the screw axis (i.e. the nut may not pulled in the direction of $\mathbf{s}$). In the infinite-pitch case shown in Fig. 7.4c, the wrench is a pure moment (i.e. the nut may not be rotated around the axis $\mathbf{s}$).

7.3.3 Reciprocal Screw Systems

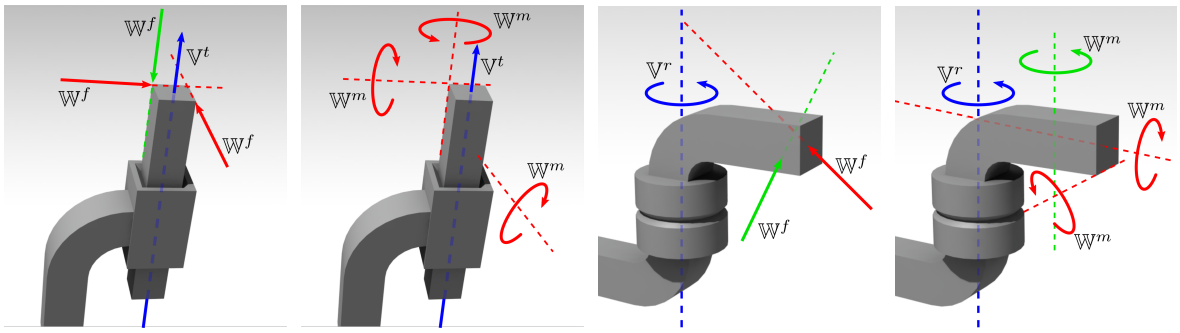
Twists and wrenches are usually used in order to characterize the kinetostatic behaviour of bodies and/or mechanisms. For a body (or a mechanism) having n DOF ($n \leq 6$), it is possible to define a set of n linearly independent twists $\mathcal{T} = \{\mathbb{v}_1 ... \mathbb{v}_n\}$ spanning the basis of the feasible translational and angular velocities achievable by the kinematic chain [207]. For serial mechanism composed of translational and revolute joints there is one twist per joint, however the rank$\left(\mathcal{T}\right)$ is at most 6, and further joints simply add kinematic redundancy. In the case of a mechanism with rank $\left(\mathcal{T}\right) = n$ DOF, there is necessarily a wrench set $\mathcal{W} = \{\mathbb{w}_1 ... \mathbb{w}_{6-n}\}$ containing $6 - n$ wrenches which constrain the twist of the mechanism to remain within $\mathcal{T}$. These wrenches are defined such that they produce no infinitesimal power if applied to the mechanism, i.e. any wrench $\mathbb{w}$ belonging to $\mathcal{W}$ must satisfy the condition

$$\mathbb{v} \circ \mathbb{w} = 0 \quad \forall \mathbb{v} \in \mathcal{T},\ \forall \mathbb{w} \in \mathcal{W} \tag{7.13}$$

where the reciprocal product (denoted by the $\circ$ operator) of two screws $\$_1$ and $\$_2$ is

$$\$_1 \circ \$_2 = \left(\begin{bmatrix} \mathbf{0} & \mathbf{I}_3 \\ \mathbf{I}_3 & \mathbf{0} \end{bmatrix} \$_1 \right)^T \$_2 \tag{7.14}$$

The reciprocity conditions of screws are well studied [205], and for the types of screws used in this paper, they can be summarized as



(a) $\mathbb{v}^{\mathrm{t}} \circ \mathbb{w}^{\mathrm{f}}$ (b) $\mathbb{v}^{\mathrm{t}} \circ \mathbb{w}^{\mathrm{m}}$ (c) $\mathbb{v}^{\mathrm{r}} \circ \mathbb{w}^{\mathrm{f}}$ (d) $\mathbb{v}^{\mathrm{r}} \circ \mathbb{w}^{\mathrm{m}}$

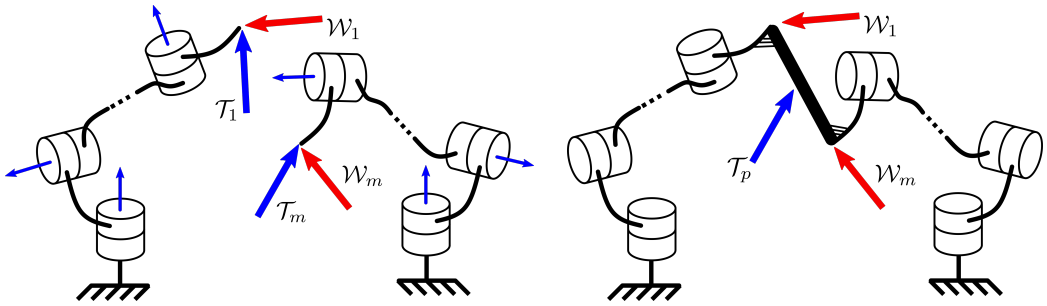
Figure 7.5 – Examples of twists and wrenches. Wrenches that produce virtual work are in green, and those that do not produce any virtual work (i.e. do not induce motion in the joint) are in red.

1. $\mathbb{v}^{\mathrm{t}}_A \circ \mathbb{w}^{\mathrm{f}}_B = 0$ if $\mathbf{s}_A$ and $\mathbf{s}_B$ are orthogonal
2. $\mathbb{v}^{\mathrm{t}}_A \circ \mathbb{w}^{\mathrm{m}}_B = 0$ for all $\mathbb{v}^{\mathrm{t}}$ and $\mathbb{w}^{\mathrm{m}}$
3. $\mathbb{v}^{\mathrm{r}}_A \circ \mathbb{w}^{\mathrm{f}}_B = 0$ if $\mathbf{s}_A$ and $\mathbf{s}_B$ intersect
4. $\mathbb{v}^{\mathrm{r}}_A \circ \mathbb{w}^{\mathrm{m}}_B = 0$ if $\mathbf{s}_A$ and $\mathbf{s}_B$ are orthogonal

These conditions include all scenarios where an effort does not create virtual work (i.e. induce infinitesimal motion) in a passive joint, and are graphically represented in Fig. 7.5. If one imagines a kinematic chain composed of three orthogonal passive prismatic joints (which would span the twist set $\mathcal{T} = \{\mathbb{v}^{\mathrm{t}}(\mathbf{x}_0), \mathbb{v}^{\mathrm{t}}(\mathbf{y}_0), \mathbb{v}^{\mathrm{t}}(\mathbf{z}_0)\}$), it can be reasoned that it would have a wrench set $\mathcal{W} = \{\mathbb{w}^{\mathrm{m}}(\mathbf{x}_0), \mathbb{w}^{\mathrm{m}}(\mathbf{y}_0), \mathbb{w}^{\mathrm{m}}(\mathbf{z}_0)\}$ constraining all moments (thus no moment exerted on the mechanism could produce a motion). Such a mechanism on the other hand would be unable to resist any force applied to the end effector, as no force could be orthogonal to all prismatic joint axes.

Thus far we have only discussed serial mechanisms, however we will later see that it is parallel mechanisms that are used to solve formation singularity problems. If we assume that a closed-loop mechanism (see Fig. 7.6) is made with m serial kinematic chains (with twist sets $\mathcal{T}_1 \dots \mathcal{T}_m$) connected to a common rigid body p (thus creating a closed-loop mechanism), the possible twists $\mathcal{T}_p$ of the common body would be difficult to directly calculate, as they would lie in the intersection of the twist sets of the m limbs

$$\mathcal{T}_p \subset \text{span}\left(\mathcal{T}_1 \cap \dots \cap \mathcal{T}_m\right) \tag{7.15}$$



(a) Finding the twist sets of each limb

(b) The platform twist is reciprocal to the wrench sets of each limb

Figure 7.6 – The procedure for analysing the mobility of the passive parallel mechanism shown in (b). First in (a) the mechanism is cut at the rigid body common to all limbs, exposing the twist sets (and therefore the wrench sets) for each individual serial kinematic chain. The platform twist may be calculated as having a null reciprocal product with all limb constraints.

Using screw theory however, we know that each limb i also exerts a set of constraints $\mathcal{W}_i$ which must respect Eq. (7.13) with regards to $\mathcal{T}_i$, and prevents the end of limb i from moving with certain twists. The wrench set of the rigid body $\mathcal{W}_p$ thus spans the union of the individual wrench sets of each serial chain [207], as the constraining efforts of each limb must be respected. By finding the constraint set

$$\mathcal{W}_p = \text{span}\left(\mathcal{W}_1 \cup ... \cup \mathcal{W}_m\right) \tag{7.16}$$

that acts on the common body attached to each limb, we can find the set of allowable motions $\mathcal{T}_p$ for that body, such that Eq. (7.13) holds true. Furthermore by recalling that the undesired singular configurations introduce an uncontrolled motion augmenting rank $\left(\mathcal{T}_p\right)$ by s additional DOF, we can find the singular conditions in the kinematic chains by looking for conditions which cause $\mathcal{W}_p$ to lose rank by s DOF. A complete example of this using the simplest possible mechanism (the same as in Fig. 7.2(c-d)) is presented in appendix C.

7.4 Virtual Kinematic Mechanisms

The “hidden robot" is a concept that was initially developed for the analysis of visual servoing singularities [208], and which was recently extended to the mobility and singularity analysis of bearing rigid formations [209]. This concept allows the representation of bearing measurements as a set of kinematic joints linked together by rigid bodies, creating a virtual

kinematic mechanism. The wrench set of this virtual mechanism is shown to be analogous to the constraints imposed by the measurement sets in the fleet of agents, and on the constraints of the robots.

7.4.1 Bearing Measurements as Kinematic Mechanisms

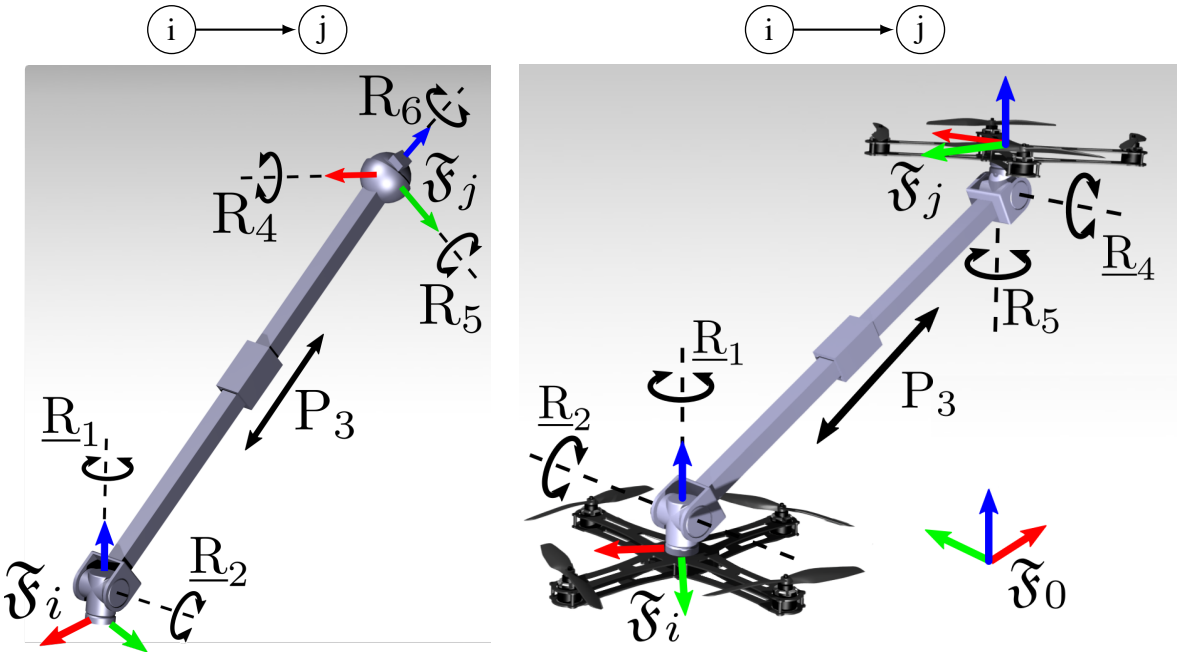
In the general case of a bearing measurement taken in $\mathfrak{F}_i$ and directed to $\mathfrak{F}_j$, the bearing measurement $\boldsymbol{\beta}_{ij}$ geometrically constrains the position of $\mathfrak{F}_j$ expressed in $\mathfrak{F}_i$ (denoted as $\mathbf{p}_{ij}$) to be

$$\mathbf{p}_{ij} = d_{ij}\boldsymbol{\beta}_{ij} \quad \forall d_{ij} > 0 \tag{7.17}$$

Moreover, a bearing measurement $\boldsymbol{\beta}_{ij}$ does not geometrically constrain the orientation of $\mathfrak{F}_j$ wrt to $\mathfrak{F}_i$. When $\mathfrak{F}_i$ and $\mathfrak{F}_j$ may both move freely in $SE(3)$, it has been shown in [210] that the geometric constraint introduced by the measurement can be represented as a virtual UPS kinematic chain[4] (Fig. 7.7a). The active U joint is centred on $\mathfrak{F}_i$ and controls the direction of the prismatic joint linking $\mathfrak{F}_i$ to $\mathfrak{F}_j$: this direction is given in $\mathfrak{F}_i$ by the bearing $\boldsymbol{\beta}_{ij}$. The combination of the U and P joints constrain $\mathfrak{F}_j$ to move on the line defined by Eq. (7.17). Additionally, the spherical joint is coincident with $\mathfrak{F}_j$, but as it is passive (i.e. its motion is neither controlled nor measured), there is no constraint on the orientation of $\mathfrak{F}_j$ relative to $\mathfrak{F}_i$.

In the case of quadrotors, which have their roll and pitch coupled to translations in the horizontal plane, the virtual mechanism accounts for the under-actuation by constraining their roll and pitch to zero [209] (the roll and pitch of the quadrotor is well estimated, thus any bearing may always be reprojected flat). This reduces the original UPS virtual kinematic chain of a bearing measurement in $SE(3)$ to a UPRR kinematic chain (Fig. 7.8b), where the active revolute joint constrains the final passive revolute joint to have a vertical axis, representing the unknown yaw of $\mathcal{A}_j$. A mutual observation between $\mathcal{A}_i$ and $\mathcal{A}_j$ leads to a UPU kinematic chain, constraining the relative yaw between the two agents. Because the singularities in the wrench set of closed-loop mechanisms (i.e. those which correspond to the singularities of the bearing formations) arise due to unconstrained motions of their passive kinematic pairs [201], all active joints may be ignored leaving only the passive joints. Thus the UPRR kinematic chain can be simplified to a PR kinematic chain, and the UPU kinematic

4. “U" indicates a universal joint, and “S" a spherical joint, where U and S are respectively composed of two and three intersecting orthogonal revolute joints. “P” indicates a prismatic (purely translational) joint. An underline indicates an active (or motorized/measured) joint, otherwise the joint is passive.



(a) Virtual UPS mechanism

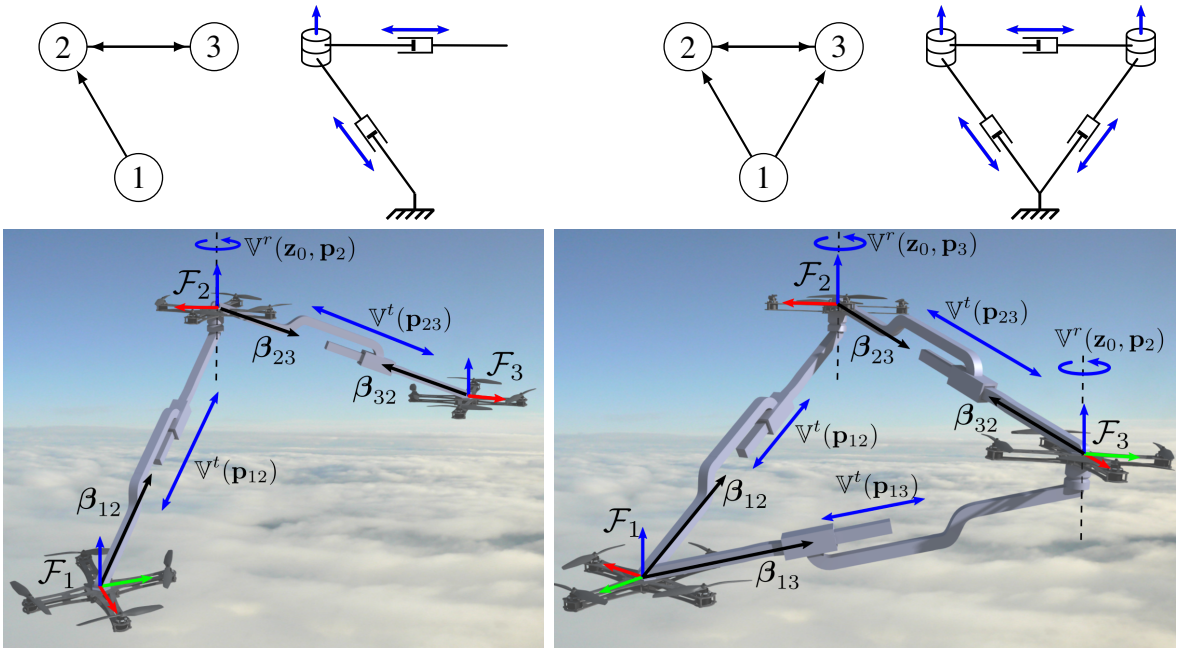
(b) Virtual UPRR mechanism

Figure 7.7 – Virtual mechanisms of directed bearing graph edges (a) on the $SE(3)$ manifold, and (b) on the $\mathbb{R}^3 \times \mathbb{S}^1$ manifolds.

chain can by simplified to a simple P joint.

7.4.2 Bearing Formations as Closed-Loop Mechanisms

In [209] it is shown how one can create virtual mechanisms from a formation graph of more than two vertices by the superposition of passive joints (and as discussed previously by treating active joints as rigid). This allows use to find the equivalent virtual kinematic mechanism for closed-looped graphs composed entirely of prismatic joints and vertical revolute joints, and which include all possible rigid bearing formations on the $\mathbb{R}^3 \times \mathbb{S}^1$ manifold. In Fig. 7.8 two examples are shown, one of an open loop formation and one of a closed-loop formation. In Fig. 7.8a, $\mathcal{A}_1$ observes $\mathcal{A}_2$, while $\mathcal{A}_2$ and $\mathcal{A}_3$ mutually observe each other. The first edge $\mathcal{E}_{12}$ is uni-directional, thus $\mathfrak{F}_2$ is constrained to $\mathfrak{F}_1$ through a PR kinematic chain. As $\mathcal{E}_{23}$ is bi-directional, $\mathfrak{F}_3$ is constrained to $\mathfrak{F}_2$ by a simple P joint. From this, we can also infer that $\mathcal{A}_3$ is constrained relative to $\mathcal{A}_1$ by a PRP kinematic mechanism. By arbitrarily fixing $\mathcal{A}_i$ in translation and orientation, we can reduce the trivial motions of the formation to an expansion of the formation about $\mathbf{p}_1$. It is then sufficient to fix a single distance (i.e. replace a P joint by a rigid member) to fix the positions of all vertices of a rigid framework. Because this naturally produces either a RP or a PR mechanism which evidently



(a) A serial formation and its virtual PRP mechanism

(b) A parallel formation and its virtual PRPRP mechanism

Figure 7.8 – The equivalent virtual kinematic mechanism for (a) a necessarily non-rigid serial formation, and (b) for a generically rigid closed-loop formation.

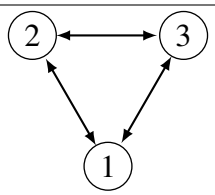
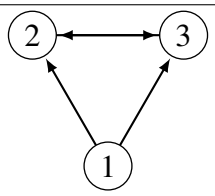
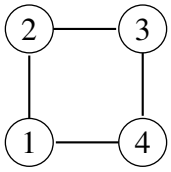
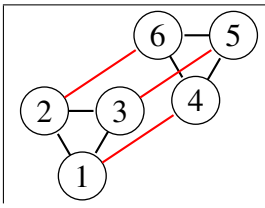
has 2 DOF, it can be seen that this formation is obviously never rigid.

If we then add a new measurement to the formation (see Fig. 7.8b) such that $\mathcal{A}_1$ sees $\mathcal{A}_3$, we produce a closed-loop graph that is generically rigid. In addition to $\mathcal{A}_3$ being constrained to $\mathcal{A}_1$ by the PR mechanism of the proceeding example (recalling that a single P joint from the PRP kinematic chain is suppressed to constrain the expansion of the formation), it is also constrained to $\mathcal{A}_1$ by a second PR kinematic chain corresponding to the added measurement. We thus have a closed-loop mechanism for which the twist and wrench set at each frame ($\mathfrak{F}_2$ and $\mathfrak{F}_3$) may be evaluated to determine its mobility (i.e. the span of the twist set of each agent after all trivial motions are constrained). Now we have presented the required tools for singularity analyses, we will next present the previously known formation singularities.

7.5 Singularities of Simple Bearing Formations

It is simple to determine whether or not a bearing formation of quadrotors is singular, all one has to do is calculate the rank of the bearing rigidity matrix. This however will only given information on whether the current configuration is indeed singular or not, without

Table 7.1 – Some bearing formation singularities that have been previously identified

Formation	Singularities
	Singular when all agents are aligned. This is widely known in the field of bearing rigidity and is trivial.
	Singular when 1. all agents are aligned 2. when graph edge $\mathcal{E}_{23}$ and $\mathcal{E}_{32}$ are vertical [116] 3. when all measurements are orthogonal to $\mathbf{z}_0$ [209]
	The bi-directional square formation is singular when agents lie on a common plane. This singularity is widely known [114], [195], [209].
	Formation composed of two distinct rigid formations are singular when all measurements (in red) joining the two rigid components are parallel [195]. This is explained as intuitive, however there is no systematic study of other possible singularities for such a system.

an indication of the behaviour at the singularity. It has indeed been remarked that only the position of the agents affects the rigidity of the formation [113], and not the yaw. If one would attempt to numerically characterize the workspace of a formation of a given graph structure, then it would require a search in $\mathbb{R}^{3|\mathcal{V}|-5}$-dimensional space for the singular loci. This is only feasible for formations of three, four, or maybe five agents, but requires simplifying conditions such as symmetry for the analysis of larger formations. Furthermore, such a simulation would only apply to that single graph structure, and would need to be redone if the graph changes. Thus the hidden robot can only be directly applied to small formations, as the analysis of large formation with multiple closed loops becomes very complicated, and therefore much of the state-of-the-art knowledge of singularities comes primarily from experienced observation and relates to relatively simple formations.

Several examples of known bearing formation singularities are included in table 7.1 and consist mostly of either directed graphs with three vertices, or bi-directional graphs with larger numbers of vertices. For a single-loop formation of four agents it has been proven multiple times that the formation is singular if all agents lie on a common plane. However for larger formations, the analysis of singularities is very sparse. When considering a graph

composed of two rigid components joined by three bi-directed measurements, it is stated as intuitive that non-trivial motions appear [195], however this is far from a complete analysis and generally complex directed formations are never considered.

7.6 Conclusion

This chapter reinforced the importance of the notion of rigidity, and in particular emphasised that it has not only a mathematical importance in the performance of control laws, but also a profound practical importance in the operation of a formation. We demonstrated the effect of singularities which make a generically rigid formation flexible, explain the physical interpretation of this flexibility, first for the more intuitive distance graphs and then for bearing graphs. The concept of the hidden robot is presented which permits the geometric evaluation of bearing formation singularities, and an introduction to screw theory is provided in order to make use of the hidden robot.

The hidden robot is limited however, in that it requires specific knowledge in the field of kinematic analysis which is generally not used in the multi-robot community. Furthermore, the complexity of multi-loop formation makes it impractical to analyse the singularities for any rigid bearing formation of more than four quadrotors. The following chapter presents one of the main contributions of this thesis, by making use of the hidden robot concept in such a way as to allow the singularity analysis of large graphs without the need for an add-hoc kinematic analysis, and for designing arbitrarily large formation graphs with fully known sets of singularities.

CHAPTER 8

IDENTIFICATION OF FORMATION SINGULARITIES

EXISTING identifications of bearing formation singularities consist of either by checking the rank of the kernel of bearing rigidity matrix in an intuitive configuration or by an analysis of the screw system of the virtual mechanism, both of which are infeasible for most large formations. This chapter shows what I believe to be the first comprehensive attempt at characterizing the singularities in bearing formations through a set-based analysis of constraints, eliminating much of the tedious work in the geometric analysis of the

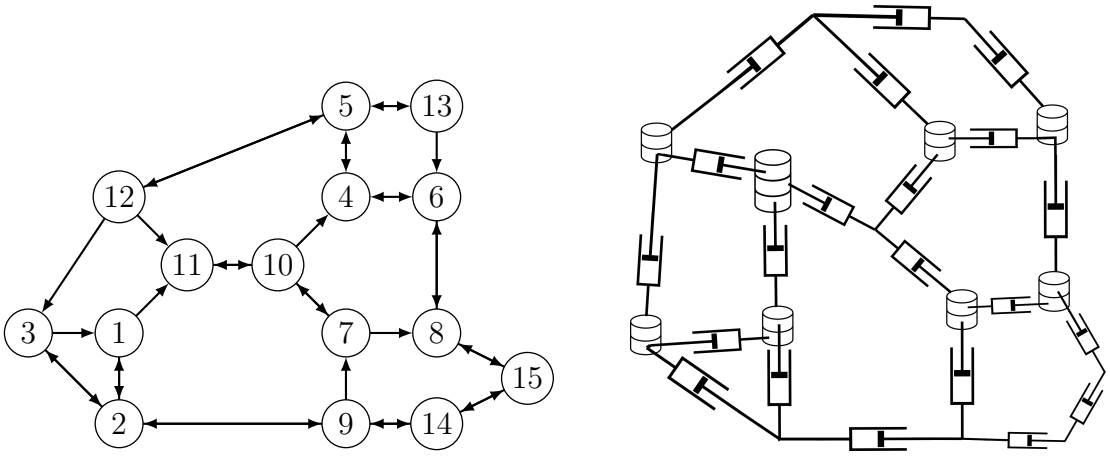
singularities of large formations.

The first section of the chapter outlines the strategy for classifying the singularities, and then the preliminary notations and edge primitives are presented. Then the different classes of singularities are shown with a complete list of all singularities for certain classifications, and a description of the limiting assumptions and cases where the analysis fails. We finally show that this method of singularity analysis can be applied to the synthesis of infinitely large formations with a conservatively known set of singularities.

8.1 Motivation

In the previous chapter, it was show how a virtual kinematic mechanism may be used to represent multi-robot formations as a set of passive kinematic links imposing the same constraints as those on the formation embedding by the robot model and inter-robot sensing. A direct application of constraint-based analysis using screw theory (or other tools such as Grassman geometry and Assur graphs) may then be performed on the virtual mechanism to determine a set of singular conditions where graph rigidity is lost. While this works well for thoroughly identifying the singularities in small formations of three or four agents, it is not intuitively scaleable. Reviewers of [209] have stated their concern that the analysis of even small formations is challenging for those in the multi-robot field who lack a strong background in multi-body kinematics, and that the analysis of larger (yet still relatively small from a multi-robot perspective) formations may be daunting even for experienced kinematicians. Indeed, classical parallel kinematic chain analysis tends to involve a multiple chains of passive joints connecting a common rigid ground link to a common rigid platform link. There have been some recent works on kinematic chains with multiple inter-connect rigid platforms [211]–[213] (which would be more representative of the virtual mechanism of a large formation), however their analysis is much more complicated that that of classical parallel chains and is currently restricted to specific kinematic architectures. As a primary advantage of decentralized multi-robot systems is their scaleability, with future applications possibly calling for hundreds or thousands of agents, an direct analysis of the singularities of such complex multi-loop mechanisms would be at best very time consuming for each individual formation, and at worst be infeasible by current practical means.

Further to the difficulty in analysing the singularities of a given formation, the extensive



(a) A relatively small bearing formation graph with 15 agents and 35 bearing measurements

(b) The equivalent virtual mechanism of the formation

Figure 8.1 – A challenging formation graph and the corresponding virtual mechanism in a generic configurations (i.e. the $\mathbb{B}$ edges are not vertical)

number of formations is dissuasive in terms of using classical singularity analysis methods. Most parallel robots are first synthesized (for practical or academic purposes) to give a some desired kinematic or dynamic properties, and singularity analysis is performed after at least defining the architecture(s) of the kinematic chain(s) linking the base to the platform. Formations however may need to self-assemble, can add or subtract edges depending on occlusions and fields of view, and more importantly have a combinatorial number of practical possibilities. If we consider a formation with $|\boldsymbol{\mathcal{V}}|$ agents, there are $p = \binom{|\boldsymbol{\mathcal{V}}|}{2}$ possible pairs of agents which may be spanned by a graph edge. Each of these p pairs may be joined by either a graph edge in one direction or the other, an bi-directional edge, or by no edge, leading to 4^p possible graphs. Due to the nature of combinatorics, this quickly leads to absurdly large numbers of possible formations. Of course, this includes many formations which will not be rigid, or even connected, but by forcing every agent to have at least a single edge joining it to every another agent, there are still 3^p possible complete graphs (almost all of which will be generically rigid). One will remark however that many of these graphs will be isomorphisms, thus sharing a common virtual mechanism and a common set of singularities. Leaving aside mathematical proofs, table 8.1 shows the number of non-isomorphic tournament graphs (complete graphs for which each edge is directed in either one or the other direction)[1] and non-isomorphic connected bi-directional graphs[2]. Not all of these formations will be rigid, however they include only a small fraction of the possible rigid graph combinations. And

1. https://oeis.org/A000568, accessed 11/10/2021
2. https://garsia.math.yorku.ca/~zabrocki/math3260w03/nall.html, accessed 11/10/2021

Table 8.1 – Number of possible graphs of different characteristics for a given number of agents. NICB stands for "Non-isomorphic connected bidirectional", and NIT for "Non-isomorphic Tournaments".

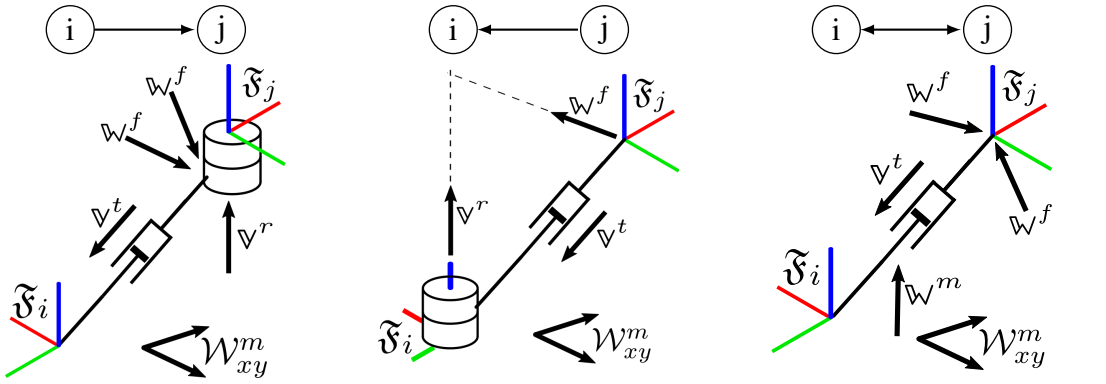
Number of Agents	**3**	**5**	**8**	**10**	**12**
Number of Pairs p	3	10	28	45	66
Directed	64	1.0 E6	7.2 E16	1.2 E27	5.4 E39
Complete Directed	27	5.9 E4	2.3 E13	2.9 E21	3.1 E31
NIT	2	12	6.8 E3	9.7 E10	1.5 E11
NICB	2	21	1.1 E4	1.1 E7	1.6 E11
Undirected Laman	1	3	608	1.1 E5	4.4 E7

finally we present in the same table the number of Laman graphs, which as stated early is a sufficient (but by no means necessary) condition to demonstrate bi-directional graph rigidity. This last could be seen as a potential set of graphs for detailed analysis, but would not be a truely thorough investigation.

The scale of the number of possible unique graphs shows the futility of cataloguing singularities using an ad-hoc analysis methodology, and emphasises instead the need for a simplified analysis of the singular configurations which can be automatically applied on-demand to a given formation graph. Furthermore, as formation control is practised primarily by a community little intersecting the kinematics community, we endeavour to develop an approach for which the kinematic analysis is pre-computed, and thus the singularities can be determined solely from the topology of the formation graph.

8.2 Preliminary Work

The first step in developing a systematic graph-based method of analysing the singular configurations of a formation is the development of a simple notation for expressing the constraints imposed on any two nodes by a graph edge. These will form a sort of set of primitives used for the rest of the analysis. Given that there exists two agents $\mathcal{A}_i$ and $\mathcal{A}_j$, and that there exists at least one graph edges connecting $\mathcal{V}_i$ and $\mathcal{V}_j$, there are three possible situations. Either the edge $\mathcal{E}_{ij} = \mathcal{V}_i \times \mathcal{V}_j$ exists, the edge $\mathcal{E}_{ji} = \mathcal{V}_j \times \mathcal{V}_i$ exists, or both edges $\mathcal{E}_{ij}$ and $\mathcal{E}_{ji}$ exist. Considering an analysis based by convention around $\mathcal{A}_i$, we can define the three aforementioned graph primitives as an out edge $\mathbb{O}$ corresponding to $\mathcal{E}_{ij}$, an in edge $\mathbb{I}$ corresponding to $\mathcal{E}_{ji}$, and a bi-directional edge $\mathbb{B}$ corresponding to the union of the two



(a) $\mathbb{O}$ equivalent kinematic chain (b) $\mathbb{I}$ equivalent kinematic chain (c) $\mathbb{B}$ equivalent kinematic chain

Figure 8.2 – The three kinematic chains for the possible graph edges

preceding types.

8.2.1 Outward Directed Edge Constraints

For an outward edge of a bearing graph (indicating a measurement of $\mathcal{A}_j$ by $\mathcal{A}_i$) we may represent the relationship between the two agents as a UPRR kinematic chain. As we are looking for conditions allowing unconstrained motions, only the passive chain is of interest [201], therefore we reduce the analysis to the PR kinematic sub-chain of the aforementioned mechanism. This results in two joint twists forming the allowable twist set

$$\mathcal{T}_{\mathbb{O}} = \left\{ \mathbb{w}^{\mathrm{t}}(\mathbf{p}_{ij}),\ \mathbb{w}^{\mathrm{r}}(\mathbf{z}_0, \mathbf{p}_j) \right\} \tag{8.1}$$

where $\mathbf{p}_{ij}$ is the line passing through $\mathbf{p}_i$ and $\mathbf{p}_j$ in an arbitrary frame of reference. This corresponds to a passive translation along the bearing direction and an passive yaw rotation of $\mathcal{A}_j$ than cannot be accounted for by the formation controller. Because the yaw of the agents does not contribute the the rigidity of the formation, we do not deal with bearings which are in each agents' local frames, but rather with spatial coordinates in an arbitrary frame on the $\mathbb{R}^3 \times \mathbb{S}^1$ manifold. This twist set has a reciprocal wrench set

$$\mathcal{W}_{\mathbb{O}} = \left\{ \mathcal{W}^m_{xy},\ \mathbb{w}^{\mathrm{f}}(\mathbf{p}^{\perp 1}_{ij}, \mathbf{p}_j),\ \mathbb{w}^{\mathrm{f}}(\mathbf{p}^{\perp 2}_{ij}, \mathbf{p}_j) \right\} \tag{8.2}$$

were $\mathcal{W}^m_{xy} = \left\{ \mathbb{w}^{\mathrm{m}}(\mathbf{x}_0), \mathbb{w}^{\mathrm{m}}(\mathbf{y}_0) \right\}$ is a wrench set containing two moments spanning a plane orthogonal to $\mathbf{z}_0$, and $\mathbb{w}^{\mathrm{f}}(\mathbf{p}^{\perp 1}_{ij}, \mathbf{p}_j)$ and $\mathbb{w}^{\mathrm{f}}(\mathbf{p}^{\perp 2}_{ij}, \mathbf{p}_j)$ are two force wrenches applied at $\mathbf{p}_j$ and spanning the plane orthogonal to $\mathbf{p}_{ij}$.

We remark that the set of moment wrenches $\mathcal{W}^m_{xy} = \left\{\mathbb{w}^{\mathrm{m}}(\mathbf{x}_0), \mathbb{w}^{\mathrm{m}}(\mathbf{y}_0)\right\}$ will be present for all edges, as these constraints are due to the underactuated robot model, and are necessary to constrain the formation state the the $\mathbb{R}^3 \times \mathbb{S}^1$ manifold. These constraints would disappear if we were to consider fully-actuated UAVS that are controllable in $SE(3)$.

8.2.2 Inward Directed Edge Constraints

This is considered an "in" edge, denoted as $\mathbb{I}$, as the graph edge $\mathcal{E}_{ji}$ enters the vertex $\mathcal{V}_i$ which corresponds to the constraints of $\mathcal{A}_j$ relative to $\mathcal{A}_i$ as imposed by measurement β_{ij} taken by $\mathcal{A}_j$. The virtual mechanism of this edge has a passive RP structure (Fig. 3b), with the twist set

$$\mathcal{T}_{\mathbb{I}} = \left\{\mathbb{w}^{\mathrm{t}}(\mathbf{p}_{ij}),\ \mathbb{w}^{\mathrm{r}}(\mathbf{z}_0, \mathbf{p}_i)\right\} \tag{8.3}$$

where $\mathbb{w}^{\mathrm{t}}(\mathbf{p}_{ij})$ is defined below Eq. (8.1) and $\mathbb{w}^{\mathrm{r}}(\mathbf{z}_0, \mathbf{p}_i)$ is a pure rotation twist around $\mathbf{z}_0$ with null translational velocity at $\mathbf{p}_i$. The wrench set constraining $\mathcal{A}_j$ to $\mathcal{A}_i$ is

$$\mathcal{W}_{\mathbb{I}} = \left\{\mathcal{W}^m_{xy},\ \mathbb{w}^{\mathrm{f}}(\mathbf{p}^{\perp 3}_{ij}, \mathbf{p}_j),\ \mathbb{w}^{\mathrm{f}}(\mathbf{z}_0 \times \mathbf{p}_{ij}, \mathbf{p}_i)\right\} \tag{8.4}$$

where $\mathbb{w}^{\mathrm{f}}(\mathbf{p}^{\perp 3}_{ij}, \mathbf{p}_j)$ is the force constraining $\mathcal{A}_j$ to continue observing $\mathcal{A}_i$ along β_{ji}. Its axis $\mathbf{p}^{\perp 3}_{ij}$ is the vector orthogonal to $\mathbf{p}_{ij}$ and intersecting the $\mathbf{z}_0$ axis passing through $\mathfrak{F}_i$. It can be expressed as

$$\mathbf{p}^{\perp 3}_{ij} = [\mathbf{p}_{ij}]_\times [\mathbf{z}_0]_\times \mathbf{p}_{ij} \tag{8.5}$$

We remark that $\mathbf{p}^{\perp 1}_{ij}$ or $\mathbf{p}^{\perp 2}_{ij}$ as used in the previous section are arbitrary vectors forming a basis orthogonal to $\mathbf{p}_{ij}$, while $\mathbf{p}^{\perp 3}_{ij}$ is a fixed vector spanned by the aforementioned basis.

8.2.3 Bi-directed Edge Constraints

This is considered as a bi-directional edge, denoted as $\mathbb{B}$, because the graph edges $\mathcal{E}_{ij}$ and $\mathcal{E}_{ji}$ can be coalesced into a single edge joining $\mathcal{V}_i$ and $\mathcal{V}_j$, with the intersection of their two twist sets

$$\mathcal{T}_{\mathbb{B}} = \mathcal{T}_{\mathbb{O}} \cap \mathcal{T}_{\mathbb{I}} \tag{8.6}$$

or equivalently, to the reciprocal twist set to the intersection of the wrench sets

$$\mathbb{v}_{\mathbb{B}} \circ \mathbb{w} = 0 \ \ \forall \mathbb{v}_{\mathbb{B}} \in \mathcal{T}_{\mathbb{B}},\ \forall \mathbb{w} \in \left(\mathcal{W}_{\mathbb{O}} \cup \mathcal{W}_{\mathbb{I}}\right) \tag{8.7}$$

Note that due to a limitation of the virtual mechanism analogy, there is a singularity in the wrench set $\mathcal{W}_{\mathbb{B}}$ when $\mathbf{p}_{ij}$ is vertical, leading to the removal of $\mathbb{w}^{\mathrm{m}}(\mathbf{z}_0)$ from $\mathcal{W}_{\mathbb{B}}$, effectively turning the $\mathbb{B}$ edge into an $\mathbb{O}$ edge. In this case, as both agents necessarily lie on the screw axis of the R joint, the exact placement of the R joint between $\mathcal{A}_i$ and $\mathcal{A}_j$ is irrelevant. This corresponds to the most general case of bearing formation analysis where the agents have no common measurable yaw reference **or** are unable to communicate their respective yaw measurements. In the case where there is a common measurable yaw available **and** the agents are able to communicate their yaw measurements to their graph neighbours, the formation graph becomes fully bi-directed and furthermore the $\mathbb{B}$ singularity is suppressed. Considering however the most general case, the twist set $\mathcal{T}_{\mathbb{B}}$ is therefore

$$\mathcal{T}_{\mathbb{B}} = \begin{cases} \left\{ \mathbb{w}^{\mathrm{t}}(\mathbf{p}_{ij}),\ \mathbb{w}^{\mathrm{r}}(\mathbf{z}_0, \mathbf{p}_j) \right\} & \text{if } \mathbf{p}_{ij} \propto \mathbf{z}_0 \\ \left\{ \mathbb{w}^{\mathrm{t}}(\mathbf{p}_{ij}) \right\} & \text{otherwise} \end{cases} \tag{8.8}$$

and the reciprocal wrenches constraining $\mathcal{A}_j$ relative to $\mathcal{A}_i$ are thus

$$\mathcal{W}_{\mathbb{B}} = \begin{cases} \left\{ \mathcal{W}_{\mathbb{O}} \right\} & \text{if } \mathbf{p}_{ij} \propto \mathbf{z}_0 \\ \left\{ \mathcal{W}_{\mathbb{O}},\ \mathbb{w}^{\mathrm{m}}(\mathbf{z}_0) \right\} & \text{otherwise} \end{cases} \tag{8.9}$$

which is similar to $\mathcal{W}_{\mathbb{O}}$, but with an additional constraint on the relative yaw so long as the edge is not vertical. It is important to remark that in all cases $\mathcal{W}_{\mathbb{O}}$ and $\mathcal{W}_{\mathbb{I}}$ are both subsets of $\mathcal{W}_{\mathbb{B}}$

$$\mathcal{W}_{\mathbb{O}} \subset \mathcal{W}_{\mathbb{B}} \tag{8.10a}$$

$$\mathcal{W}_{\mathbb{I}} \subset \mathcal{W}_{\mathbb{B}} \tag{8.10b}$$

Another noteworthy remark is that the twist set of each edge type contains at least one twist, thus every agent must have graph edges connecting it to at least two distinct agents in order to fully constrain it to the rest of the formation. This is known from various combinatorial approaches to graph rigidity analysis, but can be clearly seen by the screw theory analogy.

8.3 Classification Strategy

Due to the fact that there is currently no universal method of analysing the singularities of large multiloop mechanisms, we are unable to propose a general methodology for the complete characterization of all singularities of all formations. Instead, what we propose in this paper is a method of classifying the singularities of rigid formations, such that many formations may be completely analysed based on their graph structure. By decomposing the graph into subgraphs (as shown in Fig. 8.3 and elaborated upon in section 8.6.1), we are able to analyse the singularities of many very large graphs. Furthermore, in section 8.6.2 we show that we can design formation graphs with an almost infinite number of agents for which the singularities are completely known. To achieve this we consider two simple principles:

1. If any agent moves with respect to the rest of the formation which is fixed, the formation is singular

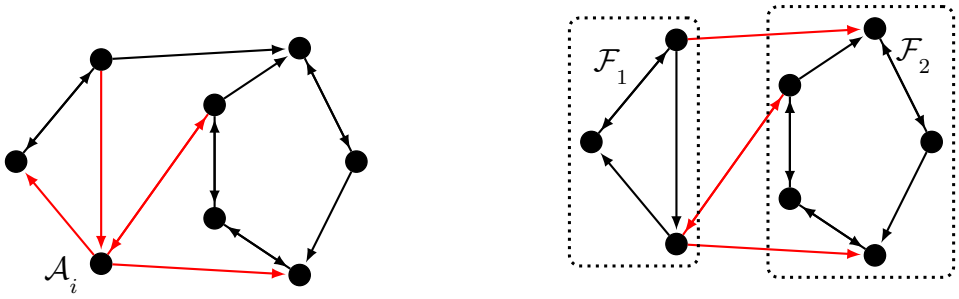
2. If any group of agents moves with respect to the rest of the formation which is fixed, the formation is singular

These two principles lead to the decomposition of the study of formation singularities into local (denoted by $\mathcal{S}^L$) and subformation (denoted by $\mathcal{S}^F$) singularities. Note that $\mathcal{S}^L$ and $\mathcal{S}^F$ are sets of geometric conditions describing the singular locii of the formation in $\mathbb{R}^3$ space, as it is known that formation rigidity is invariant to the relative yaw between the robots. These sets are used to study the formation graph decomposed into sub-graphs as shown in Fig. 8.3 and attempt to answer the questions

- **Local singularity**: Can an individual agent move with respect to it's neighbours, assuming that the neighbours are rigid?

- **Subformation singularity**: Can a group of agents move with respect to another group of agents, assuming both groups are rigid?

This type of analysis is limited in that we cannot prove that all singularities in all possible rigid graphs are detected. Indeed, there are limitations in the scope of the subformation analysis (see sections 8.5.4). We are however still able to find many singularities in those graphs for which we cannot guarentee that all singularities are identified. We do show however in section 8.6.2 that arbitrarily large formations can be built for which the



(a) Local analysis about $\mathcal{A}_i$

(b) Subformation analysis between $\mathcal{F}_1$ and $\mathcal{F}_2$

Figure 8.3 – Singularity analysis methods using local and subformation subsystems of a graph, with the analysed edges drawn in red

singularities are fully identified (i.e. all singular embeddings of the formation are spanned by a known set of $\mathcal{S}^L \cup \mathcal{S}^F$).

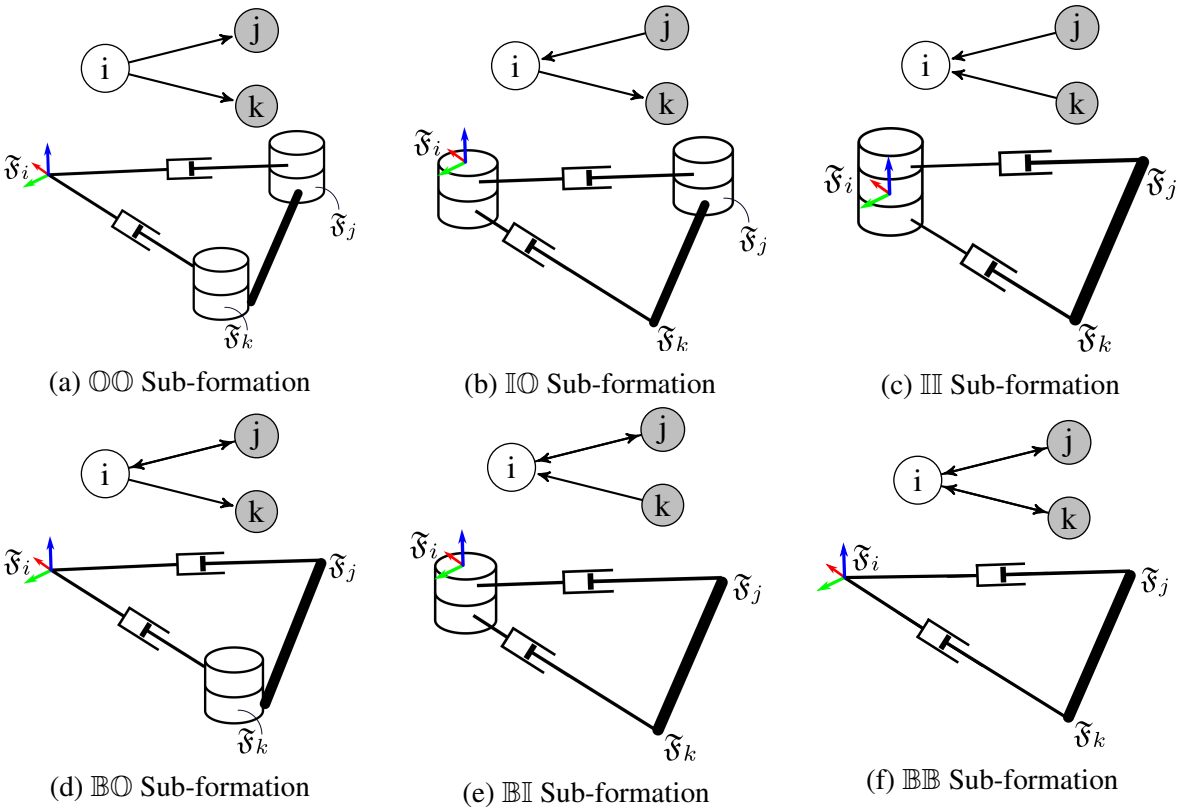
In the sections 8.4-8.5 we provide details on the local and subformation singularities types. For the most part, the results are presented and derivations using the wrench sets are not explicitly shown, however some examples are included to guide the reader should they decide to replicate the analyses.

8.4 Local Singularities

The underlying principle behind local singularities is that if any agent is flexible with respect to it's neighbours, it is a sufficient condition to lose rigidity. As such, we begin with the assumption that all the remaining agents are rigid amongst one another, which will allow the identification of all singularities arising from the embedding of any single agent and its neighbours. Applying the hidden robot methodology to the local singularity analysis, and recognizing that any relative motion between the other rigidly constrained agents is necessarily in a trivial motion, we can create a virtual mechanism for any given agent $\mathcal{A}_i$ for which each neighbour of $\mathcal{A}_i$ belongs to the ground link and $\mathcal{A}_i$ is an infinitesimally small moving platform. This allows the direct application of classical techniques for analysing singularities in parallel robots as describes in chapter 7.

8.4.1 Two-edge Local Singularities

As $\mathcal{A}_i$ may be connected to each neighbour by either either an $\mathbb{O}$ edge, an $\mathbb{I}$ edge, or a $\mathbb{B}$ edge, there are nine possible two-edge local graph structures. This will however lead to only



(a) $\mathbb{OO}$ Sub-formation (b) $\mathbb{IO}$ Sub-formation (c) $\mathbb{II}$ Sub-formation

(d) $\mathbb{BO}$ Sub-formation (e) $\mathbb{BI}$ Sub-formation (f) $\mathbb{BB}$ Sub-formation

Figure 8.4 – All possible two-edge local subformations

the six virtual mechanisms shown in Fig. 8.4 due to the kinematic irrelevance of ordering the edge types (e.g. a $\mathbb{BO}$ local graph structure has the same virtual mechanism as $\mathbb{OB}$ local graph structure). Because there are few possibilities and the virtual mechanisms are quite simple, we perform a manual analysis on all six virtual local 2-edge structures using screw theory.

In order to simplify the listing of the singular conditions we use the notation $\mathcal{S}^L_{\mathrm{XX}}$ to represent the set of geometric conditions for which the local 2-edge graph structure becomes singular. We also exploit the fact that $\mathcal{W}_{\mathbb{O}} \subset \mathcal{W}_{\mathbb{B}}$ and $\mathcal{W}_{\mathbb{I}} \subset \mathcal{W}_{\mathbb{B}}$, remarking that any subset of $\mathcal{W}$ may span at most the span of the constraints in $\mathcal{W}$. This implies $\mathcal{S}^L_{\mathbb{BB}}$ is a subset of all other $\mathcal{S}^L_{XX}$, as when $\mathcal{W}_{\mathbb{BB}}$ is insufficient to fully constrain $\mathcal{A}_i$, then there will be no subset of $\mathcal{W}_{\mathbb{BB}}$ that could constraint $\mathcal{A}_i$ any better. We thus begin with the most constrained local formation, and expand the set of singular conditions as constraints are removed from the local graph structure. A comprehensive list of all two-edge local singularities are presented in table 8.2, with the derivation of a selection of the singularities as follows.

Table 8.2 – Singularities for 2-Edge local formations of $\mathcal{A}_i$ connected to agents $\mathcal{A}_j$ and $\mathcal{A}_k$

Type	Wrenches	Singular Configuration $\mathcal{S}^{\mathcal{L}}_{\text{type}}$	Singular Twist $\mathcal{T}_{\mathcal{L}\text{type}}$
$\mathcal{L}_{\mathbb{BB}}$	$\mathcal{W}_{\mathbb{B}j} \cup \mathcal{W}_{\mathbb{B}k}$	1. $\mathcal{S}^{\mathcal{L}}_{\mathbb{BB}}$ (see Eq. (8.12))	1. $\mathcal{T}_{\mathcal{L}\mathbb{BB}}$ (see Eq. (8.13))
$\mathcal{L}_{\mathbb{BO}}$	$\mathcal{W}_{\mathbb{B}j} \cup \mathcal{W}_{\mathbb{O}k}$	1. $\mathcal{S}^{\mathcal{L}}_{\mathbb{BB}}$	1. $\mathcal{T}_{\mathcal{L}\mathbb{BB}}$
$\mathcal{L}_{\mathbb{BI}}$	$\mathcal{W}_{\mathbb{B}j} \cup \mathcal{W}_{\mathbb{I}k}$	1. $\mathcal{S}^{\mathcal{L}}_{\mathbb{BB}}$ 2. $\mathbf{p}_{ij}$ is vertical	1. $\mathcal{T}_{\mathcal{L}\mathbb{BB}}$ 2. $\mathbb{w}^{\text{r}}(\mathbf{z}_0, \mathbf{p}_i)$
$\mathcal{L}_{\mathbb{OI}}$	$\mathcal{W}_{\mathbb{O}j} \cup \mathcal{W}_{\mathbb{I}k}$	1. $\mathcal{S}^{\mathcal{L}}_{\mathbb{BI}}$ 2. $\mathbf{p}_{ij}$ and $\mathbf{p}_{ik}$ are horizontal	1. $\mathcal{T}_{\mathcal{L}\mathbb{BI}}$ 2. $\mathbb{w}^{\text{r}}(\mathbf{z}_0, \mathbf{c})$
$\mathcal{L}_{\mathbb{OO}}$	$\mathcal{W}_{\mathbb{O}j} \cup \mathcal{W}_{\mathbb{O}k}$	1. $\mathcal{S}^{\mathcal{L}}_{\mathbb{BO}}$ 2. $\mathbf{p}_{ij}$ and $\mathbf{p}_{ik}$ are horizontal 3. $\mathbf{p}_{jk}$ is vertical	1. $\mathcal{T}_{\mathcal{L}\mathbb{BO}}$ 2. $\mathbb{w}^{\text{r}}(\mathbf{z}_0, \mathbf{c})$ 3. $\mathbb{w}^{\text{r}}(\mathbf{z}_0, \mathbf{p}_j)$
$\mathcal{L}_{\mathbb{II}}$	$\mathcal{W}_{\mathbb{I}j} \cup \mathcal{W}_{\mathbb{I}k}$	1. $\mathcal{S}^{\mathcal{L}}_{\mathbb{BI}}$ 2. All configurations	1. $\mathcal{T}_{\mathcal{L}\mathbb{BI}}$ 2. $\mathbb{w}^{\text{r}}(\mathbf{z}_0, \mathbf{p}_i)$

8.4.1.1 $\mathbb{BB}$ Local Graph Structure

We begin by working through the derivation of the local singularities for a $\mathcal{L}_{\mathbb{BB}}$ subformation. The wrench set $\mathcal{W}_{\mathbb{BB}}$ constraining $\mathcal{A}_i$ to the two fixed agents $\mathcal{A}_j$ and $\mathcal{A}_k$ is spanned by

$$\mathcal{W}^{L}_{\mathbb{BB}} = \begin{cases} \mathcal{W}^{\text{m}}_{xy},\ \mathbb{w}^{\text{m}}(\mathbf{z}_0), \\ \mathbb{w}^{\text{f}}_{j1}(\mathbf{p}^{\perp 1}_{ij}, \mathbf{p}_j),\ \mathbb{w}^{\text{f}}_{j2}(\mathbf{p}^{\perp 2}_{ij}, \mathbf{p}_j),\ \mathbb{w}^{\text{f}}_{k1}(\mathbf{p}^{\perp 1}_{ik}, \mathbf{p}_k),\ \mathbb{w}^{\text{f}}_{k2}(\mathbf{p}^{\perp 2}_{ik}, \mathbf{p}_k) \end{cases} \tag{8.11}$$

which can be broken down into three orthogonal moments and two pairs of intersecting forces. Note that in the special case where both edges $\mathbf{p}_{ij}$ and $\mathbf{p}_{ij}$ are vertical, $\mathbb{w}^{\text{m}}(\mathbf{z}_0)$ disappears, otherwise due to reciprocity condition #3 there are no possible revolute singular twists. We may therefore begin by restricting our search to singular conditions allowing for translational singular twists $\mathbb{v}^{\text{t}}_s$, which from the reciprocity conditions, must be orthogonal to all forces $\mathbb{w}^{\text{f}}$. As the forces in $\mathcal{W}_{\mathbb{BB}}$ span the planes orthogonal to the edges $\mathbf{p}_{ij}$ and $\mathbf{p}_{ik}$, the singular condition permitting $\mathbb{v}^{\text{t}}_s$ must be parallel to $\mathbf{p}_{ij}$ and $\mathbf{p}_{ik}$, and thus the singularity arises when the two edges are collinear, as is well known in literature. Returning now to the special case where both edges are vertical, the wrench $\mathbb{w}^{\text{m}}(\mathbf{z}_0)$ disappears, permitting a singular revolute twist that must be both orthogonal to the remaining moments $\mathcal{W}^{\text{m}}_{xy}$ and intersect all forces. This corresponds to at pure rotation of $\mathcal{A}_i$ around the $\mathbf{z}_0$ axis passing

through $\mathbf{p}_i$ (and thus through $\mathbf{p}_j$ and $\mathbf{p}_k$). The resulting singular conditions are therefore

$$\mathcal{S}^{\mathcal{L}}_{\mathbb{BB}} = \begin{cases} 1.\ \mathbf{p}_{ij} \text{ and } \mathbf{p}_{ik} \text{ are colinear} \\ 2.\ \mathbf{p}_{ij} \text{ and } \mathbf{p}_{ik} \text{ are vertical} \end{cases} \tag{8.12}$$

with the second condition being a subset of the first and permitting an additional singular twist. The singular twist sets corresponding to these singular configurations are:

$$\mathcal{T}(\mathcal{S}^{\mathcal{L}}_{\mathbb{BB}}) = \begin{cases} 1.\ \mathbb{w}^{\mathrm{t}}_i(\mathbf{p}_{ij}) \\ 2.\ \mathbb{w}^{\mathrm{t}}_i(\mathbf{p}_{ij}),\ \mathbb{w}^{\mathrm{r}}_i(\mathbf{z}_i,\ \mathbf{p}_i) \end{cases} \tag{8.13}$$

which correspond in the first case to an unconstrained translation of $\mathcal{A}_i$ along the line $\mathbf{p}_{ij}$ and in the second case to the aforementioned singular twist as well as an additional unconstrained yaw of $\mathcal{A}_i$ relative to the other two agents. The first condition results in $\mathrm{rank}(\ker(\mathfrak{M})) = 6$, and the second condition is a locus within the first for which $\mathrm{rank}(\ker(\mathfrak{M})) = 7$.

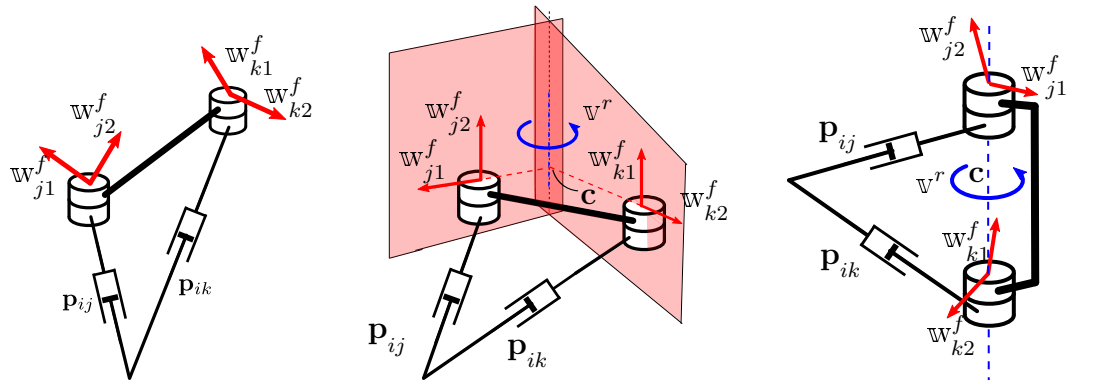
8.4.1.2 $\mathbb{BO}$ Local Graph Structure

Given that $\mathcal{W}_{\mathbb{BO}}$ is a subset of $\mathcal{W}_{\mathbb{BB}}$, we know that $\mathcal{S}^{\mathcal{L}}_{\mathbb{BO}}$ necessarily contains $\mathcal{S}^{\mathcal{L}}_{\mathbb{BB}}$. In fact, in a generic configuration, $\mathcal{W}_{\mathbb{BO}}$ is as is described in Eq. (8.14) however the $\mathbb{w}^{\mathrm{m}}(\mathbf{z}_0)$ wrench degenerates when $\mathbf{p}_{ij}$ is vertical. Thus any potential singular condition for $\mathcal{L}_{\mathbb{BO}}$ that is not a singular condition of $\mathcal{L}_{\mathbb{BB}}$ must necessarily come when $\mathbf{p}_{ij}$ is vertical. Within these search parameters, it can be quickly found that the only singular condition is when $\mathbf{p}_{ik}$ is also vertical, which is also a singular condition of $\mathcal{L}_{\mathbb{BB}}$, therefore there are no additional singularities in the $\mathcal{L}_{\mathbb{BO}}$ local formation.

8.4.1.3 $\mathbb{OO}$ Local Graph Structure

The last solved example we will give of the 2-edge local singularities is the local singularities arising when $\mathcal{A}_i$ observes two other agents, and is not observed as may occur in leader-follow type formations of simply when a new agent joins an existing formation. We note as previously that $\mathcal{W}_{\mathbb{OO}} \subset \mathcal{W}_{\mathbb{BO}}$ and thus all singular conditions in $\mathcal{S}^{\mathcal{L}}_{\mathbb{BO}}$ will be a part of $\mathcal{S}^{\mathcal{L}}_{\mathbb{OO}}$. The wrench set is

$$\mathcal{W}^{L}_{\mathbb{BB}} = \left\{\mathcal{W}^{\mathrm{m}}_{xy},\ \mathbb{w}^{\mathrm{f}}_{j1}(\mathbf{p}^{\perp 1}_{ij}, \mathbf{p}_j),\ \mathbb{w}^{\mathrm{f}}_{j2}(\mathbf{p}^{\perp 2}_{ij}, \mathbf{p}_j),\ \mathbb{w}^{\mathrm{f}}_{k1}(\mathbf{p}^{\perp 1}_{ik}, \mathbf{p}_k),\ \mathbb{w}^{\mathrm{f}}_{k2}(\mathbf{p}^{\perp 2}_{ik}, \mathbf{p}_k)\right\} \tag{8.14}$$



(a) Generic configuration (b) Singular configuration (c) Singular configuration

Figure 8.5 – The hidden robot for $\mathcal{L}_{\mathbb{OO}}$ in generic and singular configurations. Note that $\mathbb{w}^{\mathrm{f}}_{ab} \forall a \in [j,k], b \in [1,2]$ corresponds to element $\mathbb{w}^{\mathrm{f}}(\mathbf{p}^{\perp b}_{ia}, \mathbf{p}_a)$ of the force wrench basis orthogonal to $\mathbf{p}_{ia}$.

and therefore due to the absence of bidirectional measurements, there is no constraint moment about $\mathbf{z}_0$. The obvious translational singular motion comes as a result of all agents being aligned as seen previously, so we must check then for any rotational singular twists. These must be orthogonal to $\mathcal{W}_{xy}$ (and therefore around an axis parallel to $\mathbf{z}_0$) and furthermore intersect the line of action of each constraint force. In a generic configuration (shown in Fig. 8.5a) the wrench set therefore spans 6 DOF and fully constrains the system, however there are two singular configurations where a singular twist may be introduced:

1. When $\mathbf{p}_{ij}$ and $\mathbf{p}_{ik}$ lie on a horizontal plane (Fig. 8.5b). In such a case, the constraint forces orthogonal to each measurement may be decomposed into a basis with one vertical force and one horizontal force.The two horizontal forces necessarily intersect at some location expressed by vector $\mathbf{c} = [c_x \; c_y \; 0]^T$ in $\mathfrak{F}_i$. A rotational twist around the $\mathbf{z}_0$ axis and passing through this point intersects the two horizontal forces at $\mathbf{c}$ and intersects the two vertical forces at an infinite distance along the $\mathbf{z}_0$ axis.

2. When $\mathcal{A}_j$ and $\mathcal{A}_k$ are vertically aligned (expressed otherwise as when $\mathbf{p}_{jk}$ is vertical). This is shown in Fig. 8.5c, and it is easy to see that a free rotation may occur, as a twist around this axis is orthogonal to $\mathcal{W}_{xy}$ and intersects all constraint forces.

The remaining hidden robots are not fully derived for the sake of brevity, however may be analysed with a similar approach. We note in particular however, that the $\mathcal{L}_{\mathbb{III}}$ is always singular, as there is a twist $\mathbb{v}^{\mathrm{r}}(\mathbf{z}_0, \mathbf{p}_i)$ that is present in any generic configuration, there are no constraint moments about $\mathbf{z}_0$ and one may see from Fig. 8.2 that all constraint forces necessarily intersect the $\mathbf{z}_i$ axis.

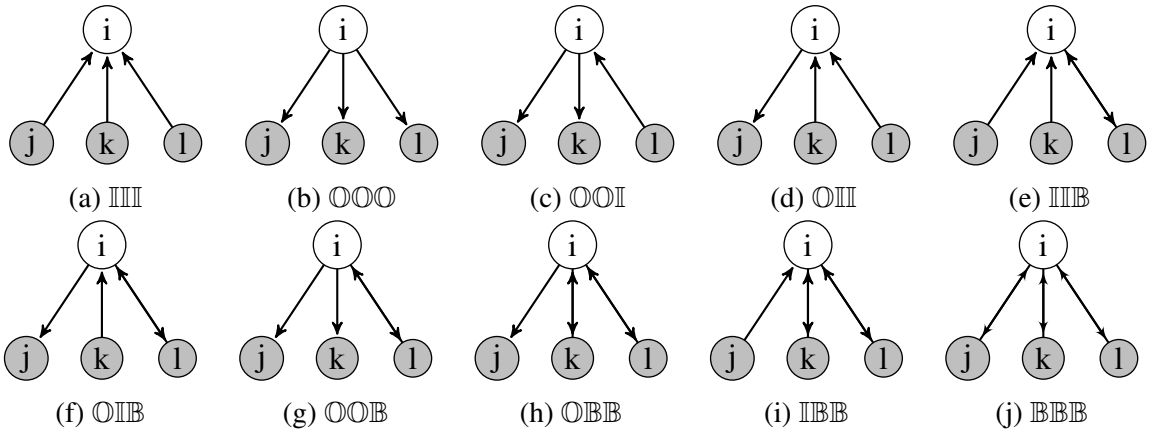


Figure 8.6 – All possible three-edge local subformations

8.4.2 Three-edge Local Singularities

The same procedure as with the two-edge local formation analysis may be used to calculate the singularities of the ten non-isomorphic three-edge local formation types shown in Fig. 8.6. Because $\mathcal{A}_i$ is more constrained, there will be fewer singularities, and they will be easier to find as any singular condition must lie in the intersection of the know known two-edge local singular sets. Considering for example the $\mathcal{L}_{\mathbb{BOI}}$ local subformation, the wrench set must span $\mathcal{W}_{\mathbb{BO}} \cup \mathcal{W}_{\mathbb{BI}} \cup \mathcal{W}_{\mathbb{OI}}$ and therefore the singular conditions must lie within $\mathcal{S}^{\mathcal{L}}_{\mathbb{BO}} \cap \mathcal{S}^{\mathcal{L}}_{\mathbb{BO}} \cap \mathcal{S}^{\mathcal{L}}_{\mathbb{BO}}$ which significantly reduces the difficulty in searching for reciprocal twists. Of the ten local formations, nine are generically rigid and thus it may be interesting to analyse their singularities. The $\mathcal{L}_{\mathbb{III}}$ local subformation type is not rigid which can be confirmed by checking that all wrenches added by the third edge are reciprocal to the singular twist $\mathbb{w}^{\mathrm{r}}(\mathbf{z}_0, \mathbf{p}_i)$ which exists in all configurations of the $\mathcal{L}_{\mathbb{III}}$ local subformation. We do not go detail all the singularities of the other nine 3-edge local formation types which may all be found in table 8.3, however the $\mathcal{L}_{\mathbb{BBB}}$ local singularities are presented as it is the simplest and the $\mathcal{L}_{\mathbb{OOO}}$ local singularities as they are the most complex.

8.4.2.1 $\mathbb{BBB}$ Local Graph Structure

While a complete wrench analysis of this system isn't complicated, the problem is simplified even more by recognizing that $\mathcal{S}^{\mathcal{L}}_{\mathbb{BBB}} \subset \mathcal{S}^{\mathcal{L}}_{\mathbb{BB}} \cap \mathcal{S}^{\mathcal{L}}_{\mathbb{BB}} \cap \mathcal{S}^{\mathcal{L}}_{\mathbb{BB}}$. It is then trivial to

Table 8.3 – Singularities of all 3-Edge local formations with $\mathcal{A}_i$ connected to agents $\mathcal{A}_j$, $\mathcal{A}_k$ and $\mathcal{A}_l$

Type	Singular Configuration $\mathcal{S}^{\mathcal{L}}_{\text{type}}$	Singular Twist $\mathcal{T}_{\mathcal{L}\text{type}}$
$\mathcal{L}_{\mathbb{BBB}}$	$\mathcal{S}^{\mathcal{L}}_{\mathbb{BBB}}$	$\mathcal{T}_{\mathcal{L}\mathbb{BBB}}$
$\mathcal{L}_{\mathbb{BBO}}$	$\mathcal{S}^{\mathcal{L}}_{\mathbb{BBB}}$	$\mathcal{T}_{\mathcal{L}\mathbb{BBB}}$
$\mathcal{L}_{\mathbb{BBI}}$	1. $\mathcal{S}^{\mathcal{L}}_{\mathbb{BBB}}$ 2. $\mathbf{p}_{ij}$ and $\mathbf{p}_{ik}$ are vertical	1. $\mathcal{T}_{\mathcal{L}\mathbb{BBB}}$ 2. $\mathbb{v}^{\text{r}}(\mathbf{z}_0, \mathbf{p}_i)$
$\mathcal{L}_{\mathbb{BOO}}$	$\mathcal{S}^{\mathcal{L}}_{\mathbb{BBB}}$	$\mathcal{T}_{\mathcal{L}\mathbb{BBB}}$
$\mathcal{L}_{\mathbb{BOI}}$	$\mathcal{S}^{\mathcal{L}}_{\mathbb{BBI}}$	$\mathcal{T}_{\mathcal{L}\mathbb{BBI}}$
$\mathcal{L}_{\mathbb{BII}}$	1. $\mathcal{S}^{\mathcal{L}}_{\mathbb{BBI}}$ 2. $\mathbf{p}_{ij}$ is vertical	1. $\mathcal{T}_{\mathcal{L}\mathbb{BBI}}$ 2. $\mathbb{v}^{\text{r}}(\mathbf{z}_0, \mathbf{p}_i)$
$\mathcal{L}_{\mathbb{OOO}}$	1. $\mathcal{S}^{\mathcal{L}}_{\mathbb{BBB}}$ 2. $\mathcal{A}_{i,j,k,l}$ lie on a common horizontal circle 3. $\mathcal{A}_{j,k,l}$ are vertical	1. $\mathcal{T}_{\mathcal{L}\mathbb{BB}}$ 2. $\mathbb{v}^{\text{r}}(\mathbf{z}_0, \mathbf{c})$ 3. $\mathbb{v}^{\text{r}}(\mathbf{z}_0, \mathbf{c})$
$\mathcal{L}_{\mathbb{OOI}}$	$\mathcal{S}^{\mathcal{L}}_{\mathbb{BBI}}$	$\mathcal{T}_{\mathcal{L}\mathbb{BBI}}$
$\mathcal{L}_{\mathbb{OII}}$	$\mathcal{S}^{\mathcal{L}}_{\mathbb{BII}}$	$\mathcal{T}_{\mathcal{L}\mathbb{BII}}$
$\mathcal{L}_{\mathbb{III}}$	1. $\mathcal{S}^{\mathcal{L}}_{\mathbb{BBB}}$ 2. All configurations	1. $\mathcal{T}_{\mathcal{L}\mathbb{BBB}}$ 2. $\mathbb{v}^{\text{r}}(\mathbf{z}_0, \mathbf{p}_i)$

show that the singular conditions are

$$\mathcal{S}^{\mathcal{L}}_{\mathbb{BBB}} = \begin{cases} 1.\ \mathbf{p}_{ij},\ \mathbf{p}_{ik}, \text{ and } \mathbf{p}_{il} \text{ are colinear} \\ 2.\ \mathbf{p}_{ij},\ \mathbf{p}_{ik} \text{ and } \mathbf{p}_{il} \text{ are vertical} \end{cases} \tag{8.15}$$

which result in the unconstrained motions

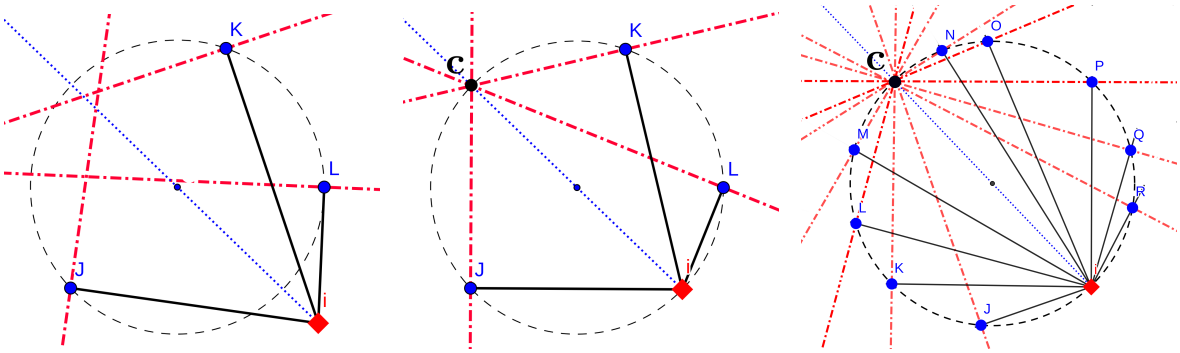
$$\mathcal{T}(\mathcal{S}^{\mathcal{L}}_{\mathbb{BBB}}) = \begin{cases} 1.\ \mathbb{v}^{\mathrm{t}}_{i}(\mathbf{p}_{ij}) \\ 2.\ \mathbb{v}^{\mathrm{t}}_{i}(\mathbf{p}_{ij}),\ \mathbb{v}^{\mathrm{r}}_{i}(\mathbf{z}_{i},\ \mathbf{p}_{i}) \end{cases} \tag{8.16}$$

We remark however that as both these singularities come with the requirement of co-linearity, there are additional measurement problems that would come with such a configuration. As bearings are primarily measured by optical devices and one agent will necessarily obscure another when there are three co-linear points observed by $\mathcal{A}_i$ this formation configuration is fairly unrealistic and should intuitively be avoided even if it were not singular.

8.4.2.2 $\mathbb{OOO}$ Local Graph Structure

It is evident by now that the set of singular conditions of the $\mathcal{L}_{\mathbb{OOO}}$ local formation includes $\mathcal{S}^{\mathcal{L}}_{\mathbb{BBB}}$. The other singular twist $\mathbb{v}^{\mathrm{r}}(\mathbf{z}_0, \mathbf{c})$ of this local subformation's two-edge subcomponents ($\mathcal{S}^{\mathcal{L}}_{\mathbb{OO}}$) occurs only when the two measurements are horizontal or when the two observed agents are vertical. We may thus restrict our search of singularities to a case where all agents lie on a horizontal plane and a case when all observed agents are vertically aligned.

1. In the first case when all agents lie an a common horizontal plane, all subcomponents singularities must have the same singular twist when expressed in $\mathfrak{F}_i$. It can be seen in Fig. 8.7a that if the agents are arbitrarily located on the horizontal plane, than the center of rotation (i.e. the intersection of the pair of horizontal constraint forces) of each pair of edges will be distinct, and thus the remaining measurement is sufficient to constrain the singular motion which is reciprocal to the first two measurements. However when all agents (including $\mathcal{A}_i$) lie on a circle, Thales' theorem (which states that *any right angle triangle inscribes within a circle has an edge bisecting the circle through its center*) proves that all horizontal constraint forces will intersect at a common point on the circle, diametrically opposite to the position of $\mathcal{A}_i$. This is represented in



(a) A non-singular configuration (b) A singular configuration (c) A singular configuration

Figure 8.7 – Singularity analysis of the $\mathbb{OOO}$ (a,b) and $\mathbb{O}^n$ (c) type local subformations on the horizontal plane. The red square is $\mathcal{A}_i$, the blue circles denote agents $\mathcal{A}_a \forall a$, and the red lines represent the force component of $\mathcal{W}_{\mathbb{O}}$ that lies in the horizontal plane. The other force components of the $\mathcal{W}_{\mathbb{O}}$ are orthogonal to the plane and pass through each blue dot.

Fig. 8.7b for the three-edge local formation and the singular twist can be represented as $\mathbb{w}^{\mathrm{r}}(\mathbf{z}_0, \mathbf{c})$.

2. In the second case, it ise simple to recognize that if all observed agents lie on a vertical line passing through a point $\mathbf{c}$, the axis of twist $\mathbb{w}^{\mathrm{r}}(\mathbf{z}_0, \mathbf{c})$ will intersect all constraint forces and is orthogonal to $\mathcal{W}_{xy}$ and it is thus a singular configuration.

Thales' theorem allows us to generalize the first unique singular condition of a single agent $\mathcal{A}_i$ measuring $n > 1$ other agents on a horizontal plane as shown by Fig. 8.7c for a 9-edge local formation. When all agents lie on a circle there is an unconstrained rotation $\mathbb{w}^{\mathrm{r}}(\mathbf{z}_0, \mathbf{c})$ of $\mathcal{A}_i$ about point $\mathbf{c}$ which is diametrically opposite across the circle.

8.4.3 Higher-edge Local Singularities

As an agent may be connected to an unlimited number of other agents, there are an infinite number of possible local formation architectures. Recalling however that locally singular conditions arise from the degeneration of $\mathcal{W}_i$, we remark that as the number of agents in $\mathcal{L}_i$ grows, the rank of $\mathcal{W}_i$ becomes less likely to degenerate as it is spanned by a greater number of individual wrenches. We can in fact show that there is a closed set of local singular configurations for the infinite number of local formation architectures.

Assuming $\mathcal{A}_i$ is connected to m other agents and each graph edge $\mathcal{E}_{ia} \forall a \in [1, m]$ is associated with a wrench set $\mathcal{W}_a$, the local wrench set $\mathcal{W}_i$ constraining $\mathcal{A}_i$ to the rest of the formation is given by Eq. (7.16). Continuing to add edges connecting $\mathcal{A}_i$ to more and more

agents, we further constrain the system. Any degeneracy of the augmented constraint system must then lie within the singularity space of the original system as well as that of the new constraints. The singularity set $\mathcal{S}_i^{\mathcal{L}}$ of the local formation of $\mathcal{A}_i$ containing m agents then becomes the intersection of all possible combinations of the two edge singularity sets

$$\mathcal{S}_i^{\mathcal{L}} = \mathcal{S}_{12}^{\mathcal{L}} \cap \mathcal{S}_{13}^{\mathcal{L}} \cap \ldots \cap \mathcal{S}_{1m}^{\mathcal{L}} \cap \ldots \cap \mathcal{S}_{(m-1)m}^{\mathcal{L}} \tag{8.17}$$

which is a closed (and quite small for large m values) set.

As agents with many edges are less likely to become locally singular, there is little interest in explicitly presenting each condition further. It is simple to extrapolate any local formation from the results of table 8.3 by applying the following rules to any local formation of type $\mathcal{L}_{\mathbb{B}^a\mathbb{O}^b\mathbb{I}^c}$ with a $\mathbb{B}$ edges, b $\mathbb{O}$ edges and c $\mathbb{I}$ edges ($m = a + b + c \geq 3$):

- **Singularities of all local formations of type $\mathcal{L}_{\mathbb{B}^a\mathbb{O}^b\mathbb{I}^c}$:**
  1. All edges are co-linear
  2. All $\mathbb{B}$ edges and all $\mathbb{O}$ edges are vertical
- **Singularities of local formations of type $\mathcal{L}_{\mathbb{O}^m}$**
  1. Agents $\mathcal{A}_i$ and $\mathcal{A}_1 \cdots \mathcal{A}_m$ lie on a common horizontal circle (see Fig. 8.7c)
  2. Agents $\mathcal{A}_1 \cdots \mathcal{A}_m$ lie on a common vertical axis.

All local singularities are sufficient conditions for the formation to lose rigidity. However as the analysis is predicated on a potentially flawed hypothesis (that all agents in $\mathcal{L}_i$ other than $\mathcal{A}_i$ are rigid), the lack of local singularities is only a necessary condition to demonstrate rigidity. As such, in the following section we extend the analysis to distinct groups of agents within a given formation i.e. the analysis of subformation singularities.

8.5 Subformation Singularities

Subformation singularities (Fig. 8.3b) analyse the interaction between two presumably rigid subformations $\mathcal{F}_A$ and $\mathcal{F}_B$ of the formation $\mathcal{F}_{AB}$. This analysis is in fact a superset of the local singularity analysis, which arise when $\mathcal{F}_A$ or $\mathcal{F}_B$ contains only a single agent. As such, here we only consider the untreated case of both subformations containing more than

one agent. In this section, the identification of subformations within a larger formation is not discussed as it depends on the specific formation being studied, however it is discussed later on in section 8.6. We also deal only with subformations that are rigidly connected, and thus we do not analyse the trivial case of two subformations connected by a single edge, which will of course not constrain the relative distance between the formations, nor their relative size.

To analyse these singularities, we fix the agents of $\mathcal{F}_A$ in place and assume that $\mathcal{F}_A$ and $\mathcal{F}_B$ are intrinsically rigid. This will fully constraint the trivial motions of $\mathcal{F}_{AB}$, and the fixed agents of $\mathcal{F}_A$ become the "ground link" of a classical parallel kinematic chain. The unfixed subformation $\mathcal{F}_B$ may then modelled only by those of its agents interacting with $\mathcal{F}_A$. These agents are constrained in space by the edges connecting the two subformations, but are also subject to constraints within their own subformation, thus are modelled as being connected by prismatic joints allowing only a scaling of $\mathcal{F}_B$. As the trivial motions of $\mathcal{F}_{AB}$ are fixed by $\mathcal{F}_A$, any mobility of any agent in $\mathcal{F}_B$ is necessarily singular.

8.5.1 Two-edge Subformation Singularities

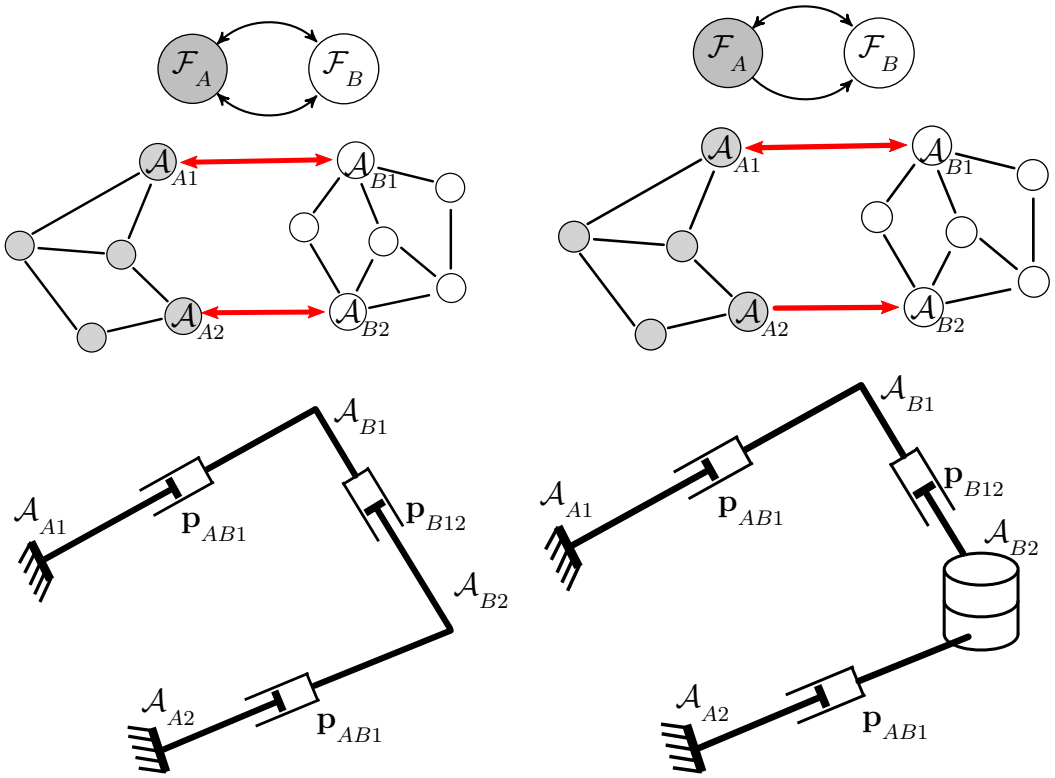
Considering the case where $\mathcal{F}_A$ and $\mathcal{F}_B$ are connected by two edges as in Fig. 8.8, there are only two possible rigid two-edge subformation types: $\mathcal{F}_{\mathbb{BB}}$ and $\mathcal{F}_{\mathbb{BO}}$, where the two subformations are connected by either two bi-directional edges, or by a bidirectional edge and a directed edge. The subformation type $\mathcal{F}_{\mathbb{BI}}$ is the same as $\mathcal{F}_{\mathbb{BO}}$, just viewed from the perspective of $\mathcal{F}_B$ instead of $\mathcal{F}_A$, and the other possible types ($\mathcal{F}_{\mathbb{OO}}$ and $\mathcal{F}_{\mathbb{OI}}$) are not generically rigid.

Starting with an analysis $\mathcal{F}_{\mathbb{BB}}$ because it's singularities are necessarily also those of $\mathcal{F}_{\mathbb{BO}}$, the wrench set constraining $\mathcal{F}_A$ to $\mathcal{F}_B$ is

$$\mathcal{W}^{\mathcal{F}}_{\mathbb{BB}} = \begin{cases} \mathcal{W}^{m}_{xy},\ \mathbb{w}^{\mathrm{m}}(\mathbf{z}_0) \\ \mathbb{w}^{\mathrm{f}}_{11}(\mathbf{p}_{A1B1} \times \mathbf{p}_{B1B2}, \mathbf{p}_{B2}) \\ \mathbb{w}^{\mathrm{f}}_{21}(\mathbf{p}^{\perp 1}_{A2B2}, \mathbf{p}_{B2}),\ \mathbb{w}^{\mathrm{f}}_{22}(\mathbf{p}^{\perp 2}_{A2B2}, \mathbf{p}_{B2}) \end{cases} \tag{8.18}$$

This wrench set degenerates under the singular conditions $\mathcal{S}^{\mathcal{F}}_{\mathbb{BB}}$:

1. **The two lines A_1B_1 and A_2B_2 intersect**: The wrench $\mathbb{w}^{\mathrm{f}}_{11}$ falls within the span of $\mathbb{w}^{\mathrm{f}}_{21}$ and $\mathbb{w}^{\mathrm{f}}_{22}$, thus $\operatorname{rank}\left(\mathcal{W}^{\mathcal{F}}_{\mathbb{BB}}\right) = 5$. The singular twist is an expansion of $\mathcal{F}_B$ about



(a) $\mathcal{F}_{\mathbb{BB}}$ (b) $\mathcal{F}_{\mathbb{BO}}$

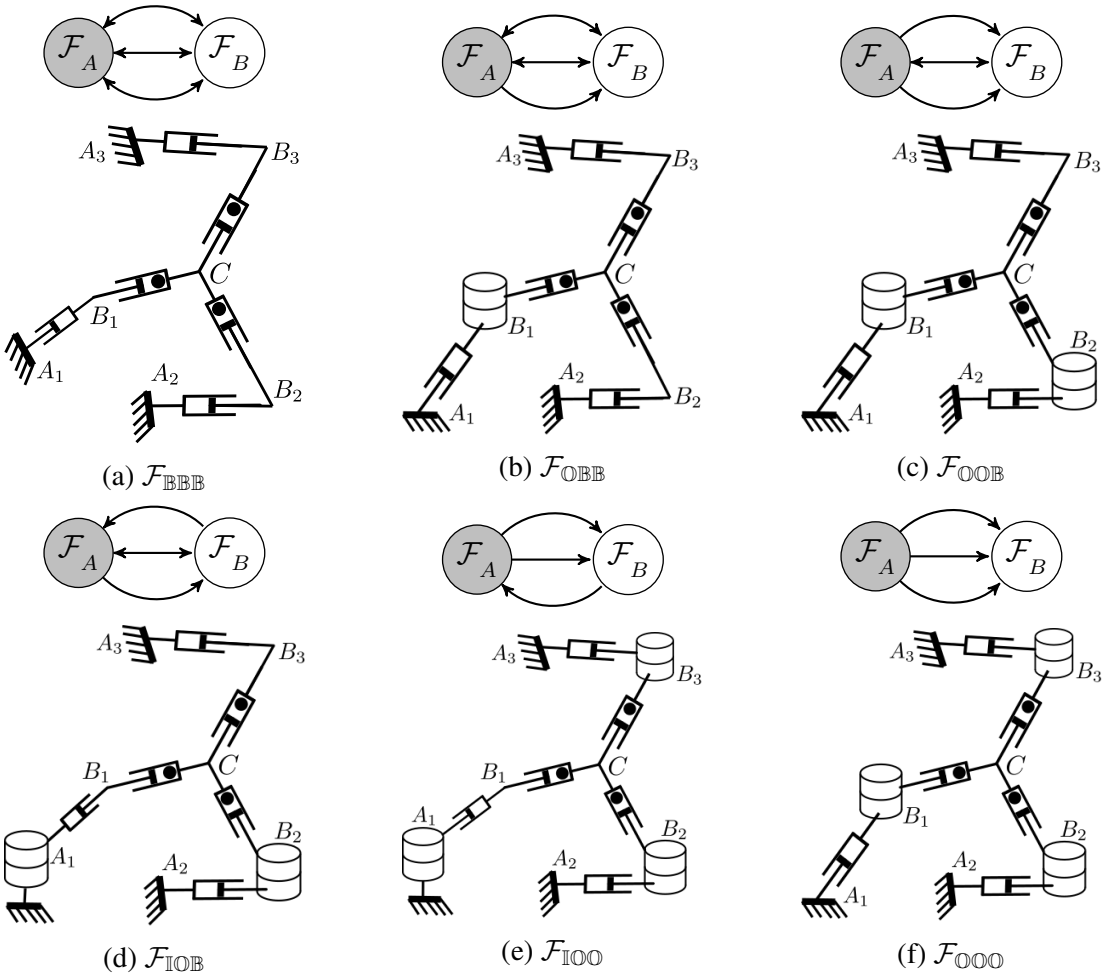
Figure 8.8 – Two subformations $\mathcal{F}_A$ and $\mathcal{F}_B$ connected by two distinct graph edges

the point of intersection P. If P lies close to infinity, then the motion is simply a translational twist $\mathbb{v}^{\mathrm{t}}(A_1B_1)$ of $\mathcal{F}_B$ with respect to $\mathcal{F}_A$

2. **The two lines are co-linear**: This is a special case of 1), resulting in $\mathcal{F}_B$ having an unconstrained twist $\mathbb{v}^{\mathrm{t}}(A_1B_1)$ with respect to $\mathcal{F}_A$, as well as an expansion about any point lying on the co-linear lines.

3. **The two lines are colinear and vertical**: This case is a special case of 2), introducing an unconstrained rotational twist $\mathbb{v}^{\mathrm{r}}(\mathbf{z}_0, A_1)$ between the two suformations.

The formation type $\mathcal{F}_{\mathbb{BO}}$ degenerates under the same conditions as $\mathcal{F}_{\mathbb{BB}}$, with an additional singularity occurring when the $\mathbb{B}$ is vertical, reducing the wrench set to that of a $\mathcal{F}_{\mathbb{OO}}$ subformation which in non-rigid. The singular motion of $\mathcal{F}_B$ with respect to $\mathcal{F}_A$ is a coupled translation along the line A_1B_1 (the $\mathbb{B}$ edge), rotation around the $\mathbf{z}_0$ axis passing through B_1, and expansion about B_1, with one total unconstrained DOF.



(a) $\mathcal{F}_{\mathbb{BBB}}$ (b) $\mathcal{F}_{\mathbb{OBB}}$ (c) $\mathcal{F}_{\mathbb{OOB}}$

(d) $\mathcal{F}_{\mathbb{IOB}}$ (e) $\mathcal{F}_{\mathbb{IOO}}$ (f) $\mathcal{F}_{\mathbb{OOO}}$

Figure 8.9 – All combinations of two subsystems A and B connected by three distinct directed graph edges. All prismatic joints containing a circle have coupled twist magnitudes. Note that the center of expansion C of subformation $\mathcal{F}_B$ can be placed at any finite point in $\mathbb{R}^3$.

8.5.2 Three-edge Subformation Singularities

There are six possible architectures of two subformations joined by three edges, all of which are shown in Fig. 8.9. and all of which are generically rigid. With these formations, we must adapt the virtual mechanism structure to account for $\mathcal{F}_B$'s ability to expand about a finite point C. To this end, we define a coupled prismatic joint, where the twist velocity is necessarily proportional to the length of the joint.

Four of the six architectures (all those which contain a $\mathbb{B}$ edge) have singularity sets which are necessarily subsets of the known two-edge rigid subformations' sets of singularities, as they contain the two-edge wrench sets along with additional constraints. We can therefore determine that

$$\mathcal{S}^{\mathcal{F}}_{\mathbb{B}_1\mathbb{B}_2\mathbb{B}_3} \subset \mathcal{S}^{\mathcal{F}}_{\mathbb{B}_1\mathbb{B}_2} \cap \mathcal{S}^{\mathcal{F}}_{\mathbb{B}_1\mathbb{B}_3} \cap \mathcal{S}^{\mathcal{F}}_{\mathbb{B}_2\mathbb{B}_3} \tag{8.19a}$$

$$\mathcal{S}_{\mathbb{OB}_1\mathbb{B}_2} \subset \mathcal{S}^{\mathcal{F}}_{\mathbb{OB}_1} \cap \mathcal{S}^{\mathcal{F}}_{\mathbb{OB}_2} \cap \mathcal{S}^{\mathcal{F}}_{\mathbb{B}_1\mathbb{B}_2} \tag{8.19b}$$

$$\mathcal{S}_{\mathbb{O}_1\mathbb{O}_2\mathbb{B}} \subset \mathcal{S}^{\mathcal{F}}_{\mathbb{O}_1\mathbb{O}_2} \cap \mathcal{S}^{\mathcal{F}}_{\mathbb{O}_1\mathbb{B}} \cap \mathcal{S}^{\mathcal{F}}_{\mathbb{O}_2\mathbb{B}} \tag{8.19c}$$

$$\mathcal{S}_{\mathbb{IOB}} \subset \mathcal{S}^{\mathcal{F}}_{\mathbb{IO}} \cap \mathcal{S}^{\mathcal{F}}_{\mathbb{IB}} \cap \mathcal{S}^{\mathcal{F}}_{\mathbb{OB}} \tag{8.19d}$$

where the edge-type subscripts are used to distinguish between distinct edges of the same type. Because these sets are all based on a contraction of known sets, we can be confident that we are able to accurately identify all singular conditions in these four subformation types.

More challenging to analyse are the two types without rigid sub-types, $\mathcal{F}_{\mathbb{IOO}}$ and $\mathcal{F}_{\mathbb{OOO}}$. While difficult to show with screw theory, we have identified the following singularities in addition to those which are calculated for the four more rigid types:

- All agents in $\mathcal{F}_A$ and $\mathcal{F}_B$ are co-planar for $\mathcal{F}_{\mathbb{IOO}}$ and $\mathcal{F}_{\mathbb{OOO}}$
- $\mathcal{F}_A$ and $\mathcal{F}_B$ are congruent for $\mathcal{F}_{\mathbb{OOO}}$: Any rigid formation $\mathcal{F}(\mathcal{G}, \mathbf{q})$ defined by a set of bearings $\boldsymbol{\beta}(\mathbf{q})$ may be transformed by an arbitrary differentiable trivial motion $\mathfrak{M}$ to a planar congruent formation $\mathcal{F}(\mathcal{G}, \mathbf{q}^*)$ such that $\boldsymbol{\beta}(\mathbf{q}) = \boldsymbol{\beta}(\mathbf{q}^*)$.

Note that these singularities have not been proven, and there may be others for the $\mathcal{F}_{\mathbb{IOO}}$ and $\mathcal{F}_{\mathbb{OOO}}$ subformation types. To completely analyse these formations, one must turn to more complex mathematical tools.

Table 8.4 – Classification of all bi-partitioned subformation singularities with two or three edges. The singular twist is often more too complex to express as a single screw when a platform is able to expanded relative to another, so the singular twist is described as either independent of coupled combinations of translations (trans.), rotations (rot.) and expansions (exp.) of $\mathcal{F}_B$ relative to $\mathcal{F}_A$.

Type	Singular Configuration $\mathcal{S}^F_{type}$	Singular Twist $\mathcal{T}^F_{type}$
$\mathbb{BB}$	1: Lines A_1B_1 and A_2B_2 intersect 2: Lines A_1B_1 and A_2B_2 are colinear 3: Lines A_1B_1 and A_2B_2 are vertical and colinear	1: Coupled trans., exp. 2: Independant trans., exp. 3: Independant trans., exp., rot.
$\mathbb{OB}$	1: $\mathcal{S}^{\mathcal{F}}_{\mathbb{BB}}$ 2: Line A_1B_1 is vertical and intersects B_2	1: $\mathcal{T}^{\mathcal{F}}_{\mathbb{BB}}$ 2: Rotation $\mathbb{w}^{\mathrm{r}}(\mathbf{z}_0, \mathbf{p}_{B2})$
$\mathbb{BBB}$	1: All lines A_iB_i intersect 2: All lines A_iB_i are colinear 3: All lines A_iB_i vertical, colinear	1: Coupled trans., exp. 2: Independant trans., exp. 3: Independant trans., exp., rot.
$\mathbb{OBB}$	1: $\mathcal{S}^{\mathcal{F}}_{\mathbb{BBB}}$ 2: $A_iB_i, i \in 2, 3$ vertical, colinear	1: $\mathcal{T}^{\mathcal{F}}_{\mathbb{BBB}}$ 2: Independant exp., rot.
$\mathbb{OOB}$	1: $\mathcal{S}^{\mathcal{F}}_{\mathbb{OBB}}$ 2: A_3B_3 vertical, intersects B_1 or B_2	1: $\mathcal{T}^{\mathcal{F}}_{\mathbb{OBB}}$ 2: Coupled trans., rot., exp.
$\mathbb{IOB}$	1: $\mathcal{S}^{\mathcal{F}}_{\mathbb{OBB}}$ 2: A_3B_3 vertical, intersects A_1, B_2	1: $\mathcal{T}^{\mathcal{F}}_{\mathbb{OBB}}$ 2: Rotation $\mathbb{w}^{\mathrm{r}}(\mathbf{z}_0, \mathbf{p}_{B2})$
$\mathbb{IOO}$	1: $\mathcal{S}^{\mathcal{F}}_{\mathbb{IOB}}$ 2: A_1 vertical colinear with $B_i, i \in 2, 3$ 3*: All agents A_i, B_i $\forall i$ lie on a common horizontal plane	1: $\mathcal{T}^{\mathcal{F}}_{\mathbb{IOB}}$ 2: Rotation $\mathbb{w}^{\mathrm{r}}(\mathbf{z}_0, \mathbf{p}_{B2})$ 3: Coupled trans., exp., rot.
$\mathbb{OOO}$	1: $\mathcal{S}^{\mathcal{F}}_{\mathbb{OOB}}$ 2: $B_i \forall i$ are vertical colinear 3*: All agents A_i, B_i $\forall i$ lie on a common horizontal plane 4*: $\mathcal{F}_A$ and $\mathcal{F}_B$ are bearing-congruent and lie on their common respective horizontal planes.	1: $\mathcal{T}^{\mathcal{F}}_{\mathbb{OOB}}$ 2: Rotation $\mathbb{w}^{\mathrm{r}}(\mathbf{z}_0, \mathbf{p}_{B2})$ 3: Coupled trans., rot., exp. 4: Coupled trans., rot., exp.

Note that an asterisk (*) denotes singularities that have not been proven with screw theory

8.5.3 Higher-edge Subformation Singularities

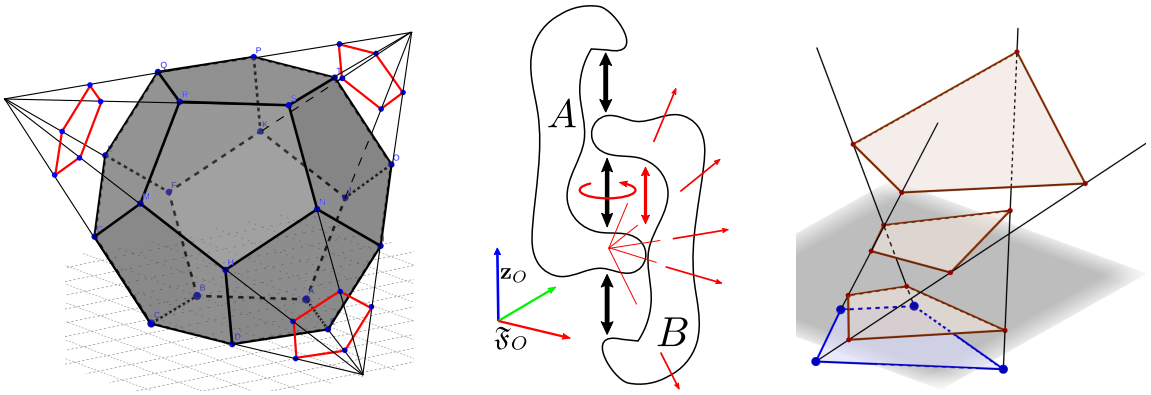
As additional edges between two subformations only adds constraints between then, we can determine (as in section 8.4C) that there is a closed set of singular conditions for all possible edge combinations. Considering subformations $\mathcal{F}_A$ and $\mathcal{F}_B$ which are joined by a $\mathbb{B}$ edges, b $\mathbb{O}$ edges, and c $\mathbb{I}$ edges, where $m = a + b + c > 3$. The singular conditions $\mathcal{S}^{\mathcal{F}}_{\mathbb{B}^a\mathbb{O}^b\mathbb{I}^c}$ are:

1. All edges intersect at a common point P
   (a) Generally, $\mathcal{F}_B$ expands with respect to $\mathcal{F}_A$ about point P.
   (b) When all edges are co-linear, $\mathcal{F}_B$ translates with respect to $\mathcal{F}_A$ and independently expands about any finite point on the line.
   (c) When all bearings are vertical and co-linear, the previous singular motion is augmented by an additional singular rotation about the co-linear bearing.
2. All $\mathbb{B}$ edges are vertical and colinear, intersecting all agents in $\mathcal{F}_B$ which are part of an $\mathbb{O}$ edge, and intersecting all agents in $\mathcal{F}_A$ which are part of an $\mathbb{I}$ edge.

There is an additional singularity $\mathcal{S}^{\mathcal{F}}_{\mathbb{O}^n}$ (and therefore of $\mathcal{S}^{\mathcal{F}}_{\mathbb{I}^n}$ as well) for formations where all edges are directed from one subformation to another. Of course, $\mathcal{S}^{\mathcal{F}}_{\mathbb{O}^n}$ contains the previously listed singular conditions, but it is also singular whenever $\mathcal{F}_A$ and $\mathcal{F}_B$ are bearing-congruent, all agents in $\mathcal{F}_A$ line on a common horizontal plane, and all agents in $\mathcal{F}_B$ lie on common horizontal plane, as is the case in Fig. 8.10c. This allows for a 1 DOF coupled translation, rotation, and expansion of $\mathcal{F}_B$ relative to $\mathcal{F}_A$.

8.5.4 Higher-partition Subformation Singularities

For some formations such as those shown in Fig. 8.11, it may be necessary to explore higher-level partitioning methods, which would be analogous to the singularities of multi-platform parallel robots. In each of the figures shown, an intuitive choice of subformations (or indeed mathematically by spectral clustering [214]) will lead to the analysis of the three rigid subformations $\mathcal{F}_1$, $\mathcal{F}_2$, and $\mathcal{F}_3$. The methodology hitherto presented would call for an analysis of the constraints between $\mathcal{F}_1$ and $\mathcal{F}_2 \cup \mathcal{F}_3$ (and the other two combinations) which would have singularities of types $\mathcal{S}^{\mathcal{F}}_{\mathbb{BB}}$ and $\mathcal{S}^{\mathcal{F}}_{\mathbb{I}^2\mathbb{O}^2}$ for the formations in figures 8.11 a) and b) respectively. We remark however, that the subformation $\mathcal{F}_2 \cup \mathcal{F}_3$ is not

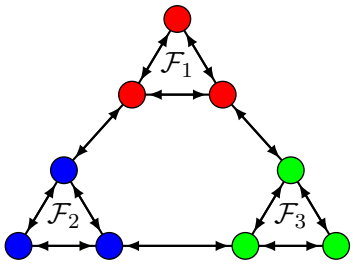


(a) Type $\mathbb{B}^5$ (b) Any type (c) Type $\mathbb{O}^n$

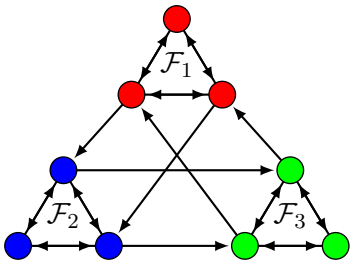
Figure 8.10 – Examples of singular motions (in red) between subformations in singular embeddings. In a) a 20-agent formation is arranged such that each agent lies on the vertex of a regular dodecahedron, and observes each of the three adjacent agents. Subfigure b) shows a general bi-partitioned formation displaying three singular degrees of freedom. Subfigure c) displays the singular motion between $\mathcal{F}_A$ (blue) and $\mathcal{F}_B$ (red), when all observations are unidirectional from $\mathcal{F}_A$ to $\mathcal{F}_B$ (i.e. of type $\mathcal{S}^F_{\mathbb{O}^4}$), and the set of observed and observing agents are congruent.

rigid, violating the hypothesis of the previous work and thus the singularities sets $\mathcal{S}^{\mathcal{F}}_{\mathbb{BB}}$ and $\mathcal{S}^{\mathcal{F}}_{\mathbb{I}^2\mathbb{O}^2}$ are only a subset of the full set of singularities. In the case of Fig. 8.11a, the formation is indeed generically bearing flexible (as determined by an analysis of the bearing rigidity matrix), and thus the singularities are of little interest. In the case of Fig. 8.11b however, the formation is generically rigid and the analysis could be potentially useful. It may be recognized that the three rigid subformations are inter-connected by only directed edges, and that in both the local singularity and the bi-partition subformation singularity analyses, a frequent singularity of directed edge sub-graphs is that all agents lie on a horizontal plane. We may therefore test and confirm numerically that if all six agents of the inner hexagon in Fig. 8.11b lie on a common horizontal plane, the formation has two singular degrees of freedom, while moving any single inner agent out of the plane restores rigidity. This however is not a rigorous analysis and could be missing many other singular configurations.

The singularity analysis of these types of formations could possibly be solved by fixing one of the three rigid subformations, and performing a kinematic analysis on the resulting virtual mechanism, similar to a multi-platform parallel robot. The analysis of these mechanisms with multiple closed kinematic loops without sharing a common ground link is currently an emerging topic in the robotics and kinematics community. Tools such as screw theory are augmented with the use of Assur graphs [211], however even without accounting for the potentially expandable platform(s) in the virtual mechanism analysis,



(a) A generically flexible graph



(b) A generically rigid graph

Figure 8.11 – Two intuitively tri-partitionable graphs, composed of three distinct rigid subformations interacting with each other. In both formations, any pair of subformations are not intrinsically rigid, and cannot be properly analysed with our method.

this is currently too complicated of an analysis to consider in this paper and is left as an open axis of future research. This section has presented all singularities arising from graphs in which distinct rigid subformations can be identified and separated by a single partition, however it leaves unsolved the singularities which arise from three or more mutually rigidifying subformations. In the next section, we present cycle singularities, which are used to compensate for the assumptions used in the previous two sections.

8.6 Applications

Up to this point we have presented lists of singular conditions for graph substructures, but in this section we present how they may be applied to a complete formation. We first show how the (possibly incomplete) analysis of singularities may be performed for non-trivial formations. We then show how arbitrarily large formations may be designed such that we are able to guarantee the complete analysis of all singularities.

8.6.1 Analysis of Large Formations

To demonstrate the application of this method of graph structure analysis for finding singular conditions for bearing formations, we describe the analysis of one specific small formation, and then discuss how it may be extended to a generic very large formation.

8.6.1.1 Methodology

We chose the formation in Fig. 8.12 based on a modified Frucht graph as our example, due to its single topological automorphism, presenting little symmetrical simplifications.

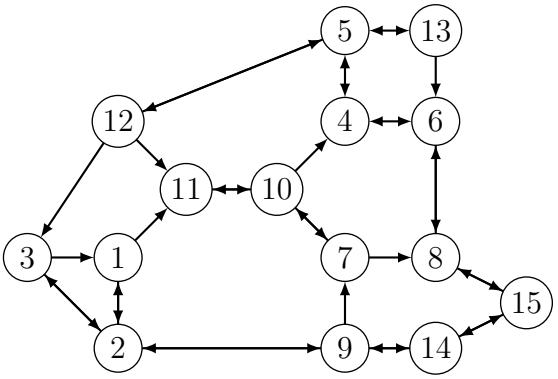


Figure 8.12 – A relatively small bearing formation graph with 15 agents and 35 bearing measurements (21 undirected edges). The label xy of a vertex referes to $\mathcal{A}_{xy}$.

This formation is composed of 15 agents and 35 directed edges (21 undirected) and is a fairly difficult formation to analyse for its size, as there are seven interconnected closed loops without any sort of serial architecture allowing a logical partitioning.

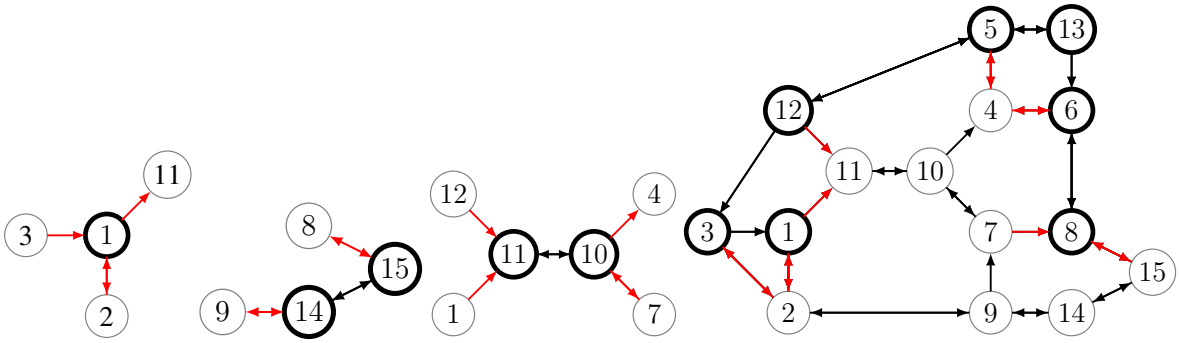
To find the local and subformation singularities of this formation, we must find all valid partitions of the formation such that it can be matched against the tables 8.2-8.4 in the previous two chapters. A simple brute force search presented in algorithm 1 is sufficient in this case, although only feasible due to the relatively small size of the formation. The

Algorithm 1: An algorithm to detect the singularities of a formation graph.

input : A directed graph $\mathcal{G}(\boldsymbol{\mathcal{V}},\boldsymbol{\mathcal{E}})$
output: A list of singular conditions $\mathcal{S}$

forall $\mathcal{V}_i \in \boldsymbol{\mathcal{V}}$ **do**
  Add $\mathcal{S}_i^L$ to $\mathcal{S}$
build undirected graph $\mathcal{G}_{\text{u}}(\boldsymbol{\mathcal{V}}, \boldsymbol{\mathcal{E}}_{\text{u}})$
for $e \leftarrow 2$ **to** $e_{max} \leq |\boldsymbol{\mathcal{E}}_u|$ **do**
  $\bar{\boldsymbol{\mathcal{E}}}$ = combinations($\boldsymbol{\mathcal{E}}_{\text{u}}$, e)
  forall $\boldsymbol{\mathcal{E}}_e \in \bar{\boldsymbol{\mathcal{E}}}$ **do**
    build cut graph $\mathcal{G}_c(\boldsymbol{\mathcal{V}}, \boldsymbol{\mathcal{E}}_{\text{u}} \backslash \boldsymbol{\mathcal{E}}_{\text{e}})$
    if $\mathcal{G}_c$.*connected_components()*=2 **then**
      if $\mathcal{G}_c.is\_connected(\mathcal{V}_i, \mathcal{V}_j) = 0 \; \forall \mathcal{E}_{ij} \in \boldsymbol{\mathcal{E}}_e$ **then**
        Add $\mathcal{S}_e^F\{\mathcal{A}_i \; \forall \mathcal{E}_{ij} \in \boldsymbol{\mathcal{E}}_{\text{e}}\}$ to $\mathcal{S}$
Result: $\mathcal{S}$

first step is simple; for each agent, check the edge directions and add the corresponding local singularities to the set of singularities for the formation. To find the subformation singularities, we must find all possible sets of edge cuts of the undirected graph $\mathcal{G}_{\text{u}}(\boldsymbol{\mathcal{V}}, \boldsymbol{\mathcal{E}}_{\text{u}})$ that result in two distinct disconnected formations, where $\boldsymbol{\mathcal{E}}_{\text{u}}$ is the set of undirected edges



(a) $\mathcal{S}^{L}_{\mathbb{RIO}}$ $\{\mathcal{A}_1\}$ (b) $\mathcal{S}^{F}_{\mathbb{B}^2}$ $\{\mathcal{A}_{14}\ \mathcal{A}_{15}\}$ (c) $\mathcal{S}^{F}_{\mathbb{RI}^2\mathbb{O}}$ $\{\mathcal{A}_{10}\ \mathcal{A}_{11}\}$ (d) $\mathcal{S}^{F}_{\mathbb{B}^5\mathbb{I}^1\mathbb{O}^2}$ $\{\mathcal{A}_1\ \mathcal{A}_3\ \mathcal{A}_5\ \mathcal{A}_6\ \mathcal{A}_8\ \mathcal{A}_{12}\ \mathcal{A}_{13}\}$

Figure 8.13 – Some examples of identified graph substructures

(i.e. $\mathcal{E}_{ij}$ is equivalent to $\mathcal{E}_{ji}$). We start by cutting all possible combinations of two edges in $\boldsymbol{\mathcal{E}}_{\text{u}}$. For each combination $\boldsymbol{\mathcal{E}}_{\text{e}}$ of two edges, the graph with the cut edges removed $\mathcal{G}_c(\boldsymbol{\mathcal{V}}, \boldsymbol{\mathcal{E}}_{\text{u}} \backslash \boldsymbol{\mathcal{E}}_{\text{e}})$ is checked using existing graph search algorithms to determine if:

1) It is bi-partitioned (i.e. has two connected components)

2) Each connected component has at least two vertices (i.e. we are not checking for local singularities)

3) Each removed edge $\mathcal{E}_{ij} \in \boldsymbol{\mathcal{E}}_{\text{e}}$ is necessary to create the bipartition (i.e. we have not removed unnecessary edges).

If the cut graph $\mathcal{G}_c$ meets these criteria, then the two components of $\mathcal{G}_c$ are a valid subformation partition that contains singularities of the type determined by the initial graph $\mathcal{G}$. We then must return and continue checking for all combination of 3 to e_{max} cut edges, where e_{max} is a threshold for ending the algorithm. In this case study, we terminated the algorithm after 12 edge cuts as no further bipartitions were detected beyond the 8-cut partition, thus we could (using post-hoc knowledge) set $\text{e}_{max} = 8$ without losing information.

8.6.1.2 Case study results

The number of sets of singularities sorted by type for the current case study is presented in table 8.5 along with the time[3] for each stage of the detection algorithm and an example of a resulting subformation (each of which comes with a set of subformation singularities). There were 15 local singularities, and 465 subformation singularities identified, some examples of which are shown in Fig. 8.13 and examples of their corresponding singular frameworks are shown in Fig. 8.14.

3. All algorithms are programmed in Python3 using the Networkx library, and run on a six-core i7 2.7 Ghz processor with 32 GB of RAM.

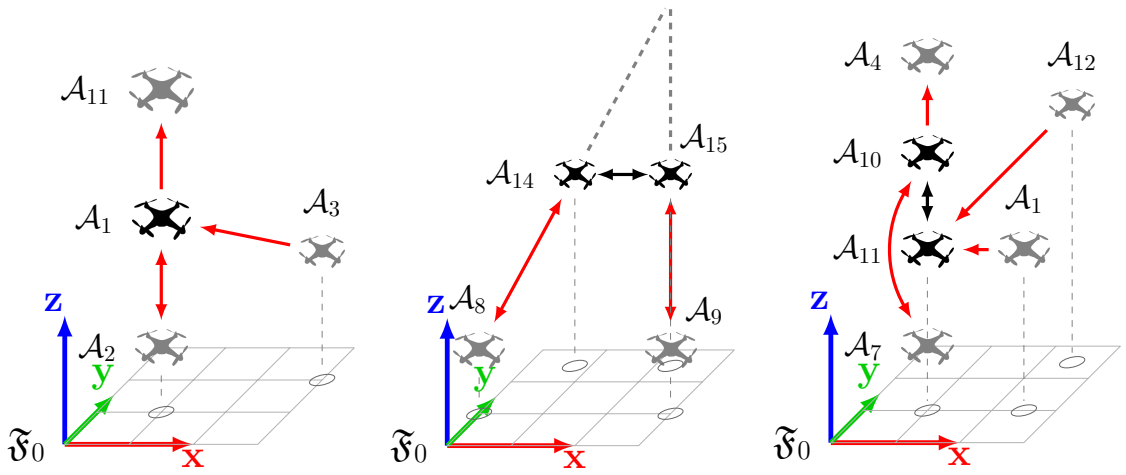
Table 8.5 – Summary of simple formation analysis with the number of graph substructures (denoted in the table by #), the search time, and an example of a graph substructure provided for each type (some of these examples are depicted in Fig. 8.13). Note that $\mathcal{S}^F_e$ (column 1) represents all subformation singularities with e connecting edges, regardless of directionality.

Type	#	Time (s)	Examples
$\mathcal{S}^L$	15	≈ 0.0	$\mathcal{S}^L_{\mathbb{BIO}} \rightarrow \mathcal{A}_1$
$\mathcal{S}^F_2$	1	0.05	$\mathcal{S}^F_{\mathbb{B}^2} \rightarrow \{\mathcal{A}_{14}\mathcal{A}_{15}\}$
$\mathcal{S}^F_3$	10	0.19	$\mathcal{S}^F_{\mathbb{B}^3} \rightarrow \{\mathcal{A}_1\mathcal{A}_2\mathcal{A}_3\mathcal{A}_{11}\mathcal{A}_{12}\}$
$\mathcal{S}^F_4$	54	0.75	$\mathcal{S}^F_{\mathbb{BI}^2\mathbb{O}} \rightarrow \{\mathcal{A}_{10}\mathcal{A}_{11}\}$
$\mathcal{S}^F_5$	106	2.8	$\mathcal{S}^F_{\mathbb{B}^3\mathbb{I}^2} \rightarrow \{\mathcal{A}_4\mathcal{A}_6\mathcal{A}_7\mathcal{A}_8\mathcal{A}_{10}\}$
$\mathcal{S}^F_6$	140	8.5	$\mathcal{S}^F_{\mathbb{B}^2\mathbb{I}^2\mathbb{O}^2} \rightarrow \{\mathcal{A}_4\mathcal{A}_5\mathcal{A}_{11}\mathcal{A}_{12}\mathcal{A}_{13}\}$
$\mathcal{S}^F_7$	110	20.6	$\mathcal{S}^F_{\mathbb{B}^2\mathbb{I}^2\mathbb{O}^3} \rightarrow \{\mathcal{A}_1\mathcal{A}_7\mathcal{A}_{10}\mathcal{A}_{11}\mathcal{A}_{12}\}$
$\mathcal{S}^F_8$	44	45.1	$\mathcal{S}^F_{\mathbb{B}^5\mathbb{I}^1\mathbb{O}^2} \rightarrow \{\mathcal{A}_1\mathcal{A}_3\mathcal{A}_5\mathcal{A}_6\mathcal{A}_8\mathcal{A}_{12}\mathcal{A}_{13}\}$
$\mathcal{S}^F_{9-12}$	0	$67-160$	None
Totals: 15 $\mathcal{S}^L$, 465 $\mathcal{S}^F$			

Any of the singular conditions from sections 8.4-8.5 being met for any of the 480 graph substructures (each of which has its own set of singular conditions) in table 8.5 is sufficient to cause a loss of rigidity in the formation. This large number of graph substructures does not however correspond to 480 distinct singular locii in the operation space of the formation as multiple graph substructures share the same geometric criteria. Let us consider for example the subformation of $\mathcal{A}_{10}$ and $\mathcal{A}_{11}$ as shown in Fig. 8.13c. The subformation singularity corresponding to this partition is $\mathcal{S}^F_{\mathbb{BI}^2\mathbb{O}}$. Referring to section 8.5.3, this is singular when all inter-subformation edges (i.e. the red edges in figures 8.13c and 8.14c) intersect at a common point. This could occurs in several geometric scenarios:

1. $\mathcal{A}_{10}$ and $\mathcal{A}_{11}$ are coincident. This is physically infeasible assuming that they are robots of non-zero size.

2. $\mathcal{A}_4$, $\mathcal{A}_7$, $\mathcal{A}_{10}$ and $\mathcal{A}_{11}$ lie on a common line (note that $\mathcal{A}_4$ and $\mathcal{A}_7$ may be replaced by $\mathcal{A}_1$ and $\mathcal{A}_{12}$). In such as case, $\mathcal{A}_{10}$ may freely translate along the line of agents, corresponding to an expansion of the subformation. This set of conditions also corresponds to the local singular conditions $\mathcal{S}^L_{\mathbb{BBO}}$ of $\mathcal{A}_{10}$, and thus the local and subformation graph substructures share the same conditions and have the same effect, thus can be considered as the same singularity. We remark in passing that if the bearings are captured by a camera as is generally the case, this singular condition



(a) $\mathcal{S}^L_{\mathbb{BIO}}\ \{\mathcal{A}_1\}$ (b) $\mathcal{S}^F_{\mathbb{B}^2}\ \{\mathcal{A}_{14}\ \mathcal{A}_{15}\}$ (c) $\mathcal{S}^F_{\mathbb{BI}^2\mathbb{O}}\ \{\mathcal{A}_{10}\ \mathcal{A}_{11}\}$

Figure 8.14 – Examples of singular frameworks for some of the graph substructures presented in figure 8.13

will necessarily result in the occlusion of at least one measurement (see Fig. 8.14c).

3. A special case of 2), where $\mathcal{A}_4$, $\mathcal{A}_7$, $\mathcal{A}_{10}$ and $\mathcal{A}_{11}$ are vertically aligned. In this case there are two new DOF (in addition to the DOF from the previous singularity); an controllable relative yaw between the subformation and the rest of the formation, and an uncontrollable yaw between $\mathcal{A}_{10}$ and $\mathcal{A}_{11}$. These can equally be described by the combination of local singularities $\mathcal{S}^L_{\mathbb{BBO}}$ and $\mathcal{S}^L_{\mathbb{BII}}$ of $\mathcal{A}_{10}$ and $\mathcal{A}_{11}$ respectively, without considering the subformation singularities at all. Numerical simulations confirm that the rank of the kernel of the bearing rigidity matrix ($\text{rank}\,(\ker(\mathbf{B}))$) in the last two scenarios is 6 and 8 respectively, corresponding to one and three added DOF as predicted.

While we clearly cannot discuss all singularities of this example in this paper, the analysis of this case study has lead to some interesting observations. *First* of all, it confirmed that our methodology is effective in detecting many sufficient conditions for bearing formation degeneracy for formations of 15 or fewer agents. *Secondly*, it was observed when analysing the results that subformation singularities arising from a large number of cut edges often (not always, as counter-examples may be presented) imply singularities arising locally, and within smaller subformations. This is expected as the singularities of the union of multiple wrench sets must lie in the intersection of the singular configurations of those individual wrench sets. This means that from a practical viewpoint, we can often limit e_{max} to reasonably low values reducing the algorithm runtime for large formations. *Thirdly*, it was remarked that while our

method always correctly predicted the existence of singularities, we could not always fully predict the resulting mobility, as sometime additional DOF would be introduces to $\ker(\mathbf{B})$ that were not accounted for. This is because our local and subformation singularity criteria are founded on the hypothesis that all agents other than those being analysed are intrinsically rigid, while any singularity breaks this assumption, potentially causing a cascade of added DOF that we (currently) cannot fully analyse without a comprehensive ad-hoc kinematic analysis.

Finally, although we can detect many singularities, there are some which went undetected. An example is when $\mathcal{A}_8$ and $\mathcal{A}_{15}$ lie on a common vertical line, which is neither a local singularity, nor a subformation singularity yet nonetheless causes rank degeneracy of $\mathbf{B}$. The reason for this is that we do not consider the complex coupling between kinematic loops, as we have not been able to formulate the analysis in a general and systematic manner. This drawback is partially mitigated however, as we show in section 8.6.2 that formations may be designed such that we can guarantee there are no unknown singularities.

8.6.1.3 A discussion on the analysis of very large formations

For a large formation, with several tens, hundreds, or thousands of agents, the analysis of its singularities may seem a daunting task. Indeed, the detection of all possible singular conditions for an arbitrary and large rigid graph is currently infeasible. By breaking the formation graph into smaller sub-graphs however, we are able to detect a large number of singularities which are, in our opinion, the most likely to occur.

Let us consider a formation of $|\boldsymbol{\mathcal{V}}|$ agents, without any specific graph structure beyond the requirement of generic bearing rigidity on the $\mathbb{R}^3 \times \mathbb{S}^1$ manifold (confirmed by checking that $\text{rank}(\ker(\mathbf{B})) = 5$ in a random embedding). For each agent $\mathcal{A}_i$, the set of local singularities $\mathcal{S}_i^{\mathcal{L}}$ can be looked up using the tables 8.2-8.3, which are easily extrapolated if needed as explained in section 8.4.3 for all possible local structures. The local singular conditions for the formation is therefore fully expressed as

$$\mathcal{S}^L = \mathcal{S}_1^L \cup \cdots \cup \mathcal{S}_{|\boldsymbol{\mathcal{V}}|}^L \tag{8.20}$$

and can be completely and easily determined for any formation of reasonably finite size (e.g. millions of agents).

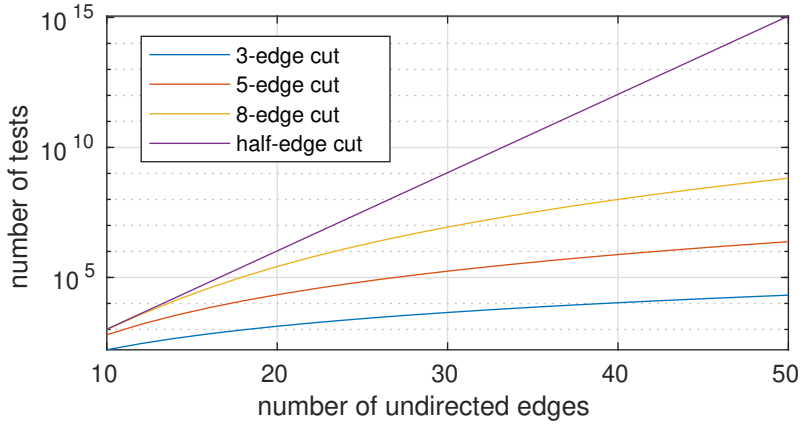


Figure 8.15 – The number of edge cut combinations to test (a strong correlation to computation time) as a function of the edge size $|\boldsymbol{\mathcal{E}}_u|$. The curves (from lowest to highest) correspond to $e_{max} = 3, 5, 8$, and $|\boldsymbol{\mathcal{E}}_u|/2$ for the maximum number of cuts. The values shown in this graph correspond to the number of evaluations of line 7 in algorithm 1, and do not account for the increase in complexity of the subsequent opperations.

Moving on to the analysis of the subformation singularities for this large formation, it can be seen from Fig. 8.15 that bi-partitioning the graph quickly becomes difficult. In fact, finding all bi-partitions in a graph is known to be an NP-hard problem [215], thus in general for large graphs it is computationally infeasible to find all possible bi-partitions. There is, however, significant work in the field of graph clustering (for data-analysis, machine learning, etc...) which provides us with time efficient heuristic-based algorithms to find prominent clusters [214] of vertices within a graph (see Fig. 8.16). These algorithms have highly-varying complexities which vary greatly depending on the method, graph, and desired partition characteristics, but can generally handle graphs with thousands of vertices (see [216] for survey).

A cluster is a group of vertices sharing a high degree of inter-connectivity compared to the connectivity with other agents. As the wrench set constraining highly connected pairs of subformations is spanned by many wrenches, the rigidity of the formation is more likely to degenerate between two subformations connected by few edges. We therefore partition the formation by seperating each cluster from the others. Given the graph is divided into c clusters, the formations may be analysed using table 8.4 (or the general extension in section 8.5.3) considering each cluster as a subformation. This will result in the set of singular subformation conditions

$$\mathcal{S}^F = \mathcal{S}_1^F \cup \cdots \cup \mathcal{S}_c^F \tag{8.21}$$

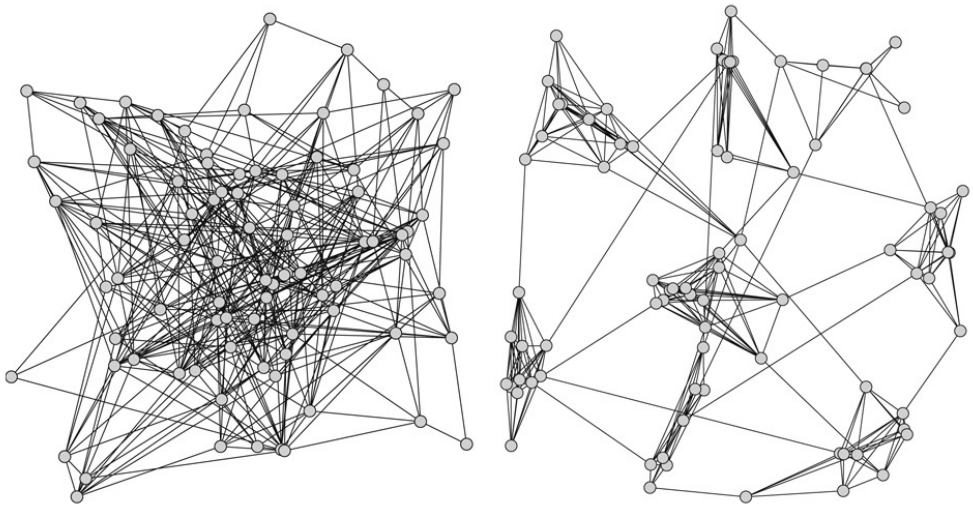


Figure 8.16 – A graph with 84 vertices and 358 edges, displayed with vertices in random positions (left), and the same graph with vertices positioned by a clustering algorithm (right) [216]

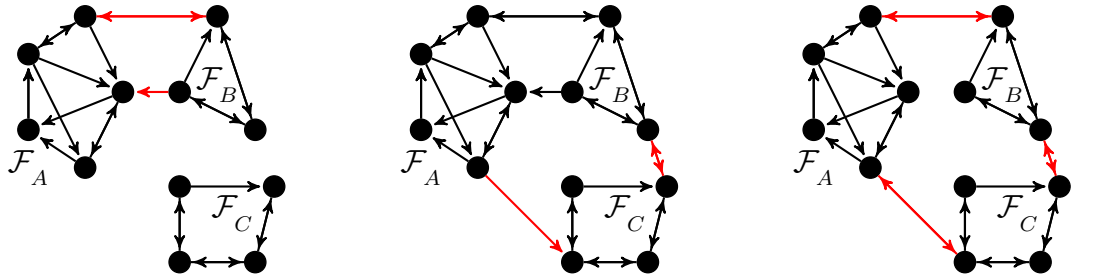
which may not contain all possible singularities, but the most likely[4] to manifest as they expose subformations connected by relatively few edges and thus spanned by relatively few wrenches. Depending on the number of vertices and edges in each cluster, another partitioning intrinsic to each cluster may be performed, either by the direct method as in section 8.6.1.1 (if the cluster is sufficiently small), or by performing another clustering.

In a formation of mobile robots, one can easily imagine that agents are more likely to observe other nearby agents that those which are far away. This gives a strong physical significance to these clusters and could potentially be used as a heuristic in a hypothetical real-time singularity analysis of formations. Furthermore this strategy of clustering may also be used to identify k-partitions ($k > 2$), in the event that their singularities are analysed later, which may be feasible in the future with advances in the analysis of multi-platform and elastic parallel robots.

8.6.2 Formation Design for Complete Knowledge of all Singularities

As discussed in the previous section, the analysis of singularities for arbitrary formations remains computationally difficult and cannot be guaranteed to detect all singular conditions. It can be shown however that arbitrarily large formation graphs with conservatively known sets of singular conditions (i.e. a set guaranteed to encompass **all** singular, and some

4. Assuming all embeddings are equally probable.



(a) Correct Operation (b) Correct Operation (c) Incorrect Operation

Figure 8.17 – Synthesis of a formation $\mathcal{F}_{ABC}$ with known singularities, from sub-formations $\mathcal{F}_A$, $\mathcal{F}_B$, and $\mathcal{F}_C$. The red edges are edges which are added to join previously seperate subformations at a given step in the proceedure.

non-singular conditions) may be designed.

8.6.2.1 Methodology

To create a large formation for which we can guarantee that all singularities lie within a known set of conditions, one must begin with two distinct intrinsically rigid formations (denoted $\mathcal{F}_A$ and $\mathcal{F}_B$) for which the singularities are well known. This is achievable for small formations composed of a basis of rigid loops (e.g. loops containing 3-4 agents with at least one or three $\mathbb{B}$ edges respectively[5]) and if desired, “formation" may refer to a single agent or two agents joined by a $\mathbb{B}$ edge. Given that the singularities of the two formations ($\mathcal{S}_A$ and $\mathcal{S}_B$) are well known, and that adding graph edges further constrains both formations, the addition of any graph edges joining $\mathcal{F}_A$ to $\mathcal{F}_B$ will only contract $\mathcal{S}_A$ and $\mathcal{S}_B$. If the added graph edges are sufficient to make $\mathcal{F}_A$ and $\mathcal{F}_B$ into a single generically rigid formation $\mathcal{F}_{AB}$, the set of singularities of $\mathcal{F}_{AB}$ is necessarily a subset of the union of $\mathcal{S}_A$, $\mathcal{S}_B$ and the singularities exposed by bi-partitioning $\mathcal{F}_{AB}$ between its two original components (denoted as $\mathcal{S}^F_{A/B}$):

$$\mathcal{S}_{AB} \subset \mathcal{S}^F_{A/B} \cup \mathcal{S}_A \cup \mathcal{S}_B \tag{8.22}$$

In the example shown in Fig. 8.17a, $\mathcal{S}^F_{A/B}$ is simply $\mathcal{S}^F_{\mathbb{BO}}$ and corresponds to the subformation partitioning exposed by cutting the red edges joining $\mathcal{F}_A$ and $\mathcal{F}_B$.

This combination of formations may then be repeated as many times as desired, so long as at every step the result is a rigid formation, and the singularities of the added formation are know. In Fig. 8.17b, the connection of $\mathcal{F}_{AB}$ to $\mathcal{F}_C$ is a rigid operation (with the singularities

5. This requirement can be confirmed by either a screw analysis or a numerical analysis of all ten possible 3-4 agent loops.

$\mathcal{S}^{F}_{AB/C}$) and $\mathcal{S}_{C}$ is known, thus the set of singularities for the formation is guaranteed to satisfy

$$\mathcal{S}_{ABC} \subset \mathcal{S}^{\mathcal{F}}_{AB/C} \cup \mathcal{S}_{AB} \cup \mathcal{S}_{C} \tag{8.23}$$

In the case of Fig. 8.17c however, we have no knowledge of the singularities from a tri-partition (see section 8.5.4, using the notation here it would be $\mathcal{S}^{F}_{A/B/C}$) thus even if we find some singularities of this formation, we cannot guarantee that all singularities lie within the set of those found.

An interesting extension of this singularity analysis by formation design is that it allows the analysis of formations with repeating structures, such as triangular lattices. In fact, as a triangular lattice may be formed by the ad infinitum addition of a new agent connected to two or more connected agents, the set of singularities will simply be a contraction of local singularities whenever a new agent is introduced.

8.6.2.2 Demonstration

To demonstrate the interest of this approach for the characterisation of formation singularities, we consider the following case study: an 18-agent, 47-edge formation created from three, four, five, and six-agent rigid formations (see Fig. 8.18). First $\mathcal{F}_{A}$ and $\mathcal{F}_{B}$ are connected to form $\mathcal{F}_{AB}$ and $\mathcal{F}_{C}$ is added to create $\mathcal{F}_{ABC}$ (the order of these two steps is interchangeable, as they both rigidly connect to $\mathcal{F}_{A}$). Finally $\mathcal{F}_{D}$ is added to create the full formation $\mathcal{F}_{ABCD}$. The undirected graph of this formation has 32 edges which, when compared to the 21 undirected edges in the example in section 8.6, should lead to many more possible partitions. We know however that edges serve to constrain the agents, therefore we should expect fewer singularities.

The set of singularities for this formation can be fully guarenteed to lie within a known set of conditions as previously explained. This set encompasses the local singularities of all agents, and some additional subformation singularities. Three of the subformation singularities correspond to the combination of rigid formations ($\mathcal{S}^{F}_{A/B}$, $\mathcal{S}^{F}_{AB/C}$, and $\mathcal{S}^{F}_{ABC/D}$) while other subformation singularities are intrinsic to the individual formations. Formations $\mathcal{F}_{B}$, $\mathcal{F}_{C}$, and $\mathcal{F}_{D}$ can in fact be created by the method we are describing, and thus we are able to guarentee that there is only one single subformation singularity in $\mathcal{F}_{D}$. We can therefore guarentee that all singularities are encompassed by 22 sets of geometric conditions corresponding to 18 local sets, three inter-subformation sets, and one intra-subformation set

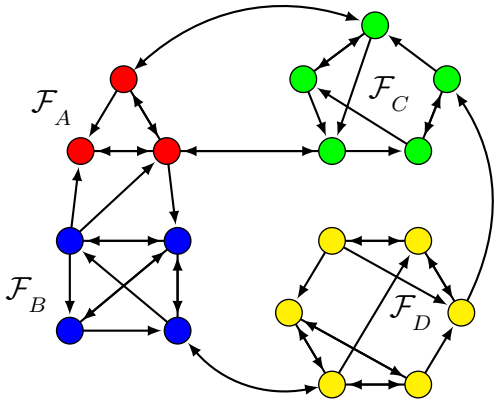


Figure 8.18 – A formation with a conservatively known set of singularities generated created by the combination of four rigid subformations $\mathcal{F}_A$, $\mathcal{F}_B$ $\mathcal{F}_C$ and $\mathcal{F}_D$ (separated by color). We remark that $\mathcal{F}_B$, $\mathcal{F}_C$ and $\mathcal{F}_D$ may be built in the manner currently being demonstrated.

(the sum of the first two columns in table 8.6).

We may also study the singularities of this formation from a more naive viewpoint, by identifying the four subformations $\mathcal{F}_A$ to $\mathcal{F}_D$ and analyse each individual subformation using the algorithm presented in section 8.6. In this case, there will be many extra singular conditions that are simply combinations of the 22 sets of conditions identified earlier. This will in fact result in 46 sets of singular conditions (the sum of the first and third columns in table 8.6). This is still small by comparison to the set of singularities identified by the direct application of algorithm 1 to the entire formation. After more than 8 hours of computation, the program ended due to a memory overflow while attempting to find all 12-cut partitions and having already detected 736 subformation singularities (and the 18 local singularities).

Because we know the formation is built by our method, we can still be confident that these sets of conditions encompass all singularities, just with more complexity as many singularities are presented which are combinations of other singularities. This demonstrates the interest in considering singularity analysis by the design approach compared to the hybrid design-analysis or the direct analysis approaches, as they express the same set of singularities as lying withing 22 sets, 46 sets, and 754 sets of singular conditions respectively.

The main limits of singularity analysis through formation design is the fact that some rigid formations (as explained in section 8.5.4) are excluded, and that the formations are required to have more edges than strictly necessary. Despite this, all formations that are created using this methodology are guaranteed to be non-singular outside of the known set of geometric conditions. In addition to providing human-interpretable information on the conditions and consequences of singularities, these conditions could potentially be included

Table 8.6 – Number of singularities in different parts of the formation shown in Fig. 8.18. The number of subformation singular conditions for a formation (e.g. $\mathcal{S}^F_C$) can be determined either by design (DE) or by algorithm 1 (AL). Both options are presented in this table, although of course the DE method will give fewer conditions.

Component	# of $\mathcal{S}^L$	# of $\mathcal{S}^F$ by DE	# of $\mathcal{S}^F$ by AL
$\mathcal{S}_A$	3	0	0
$\mathcal{S}_B$	4	0	2
$\mathcal{S}^F_{A/B}$	-	1	1
$\mathcal{S}_C$	5	0	7
$\mathcal{S}^F_{AB/C}$	-	1	1
$\mathcal{S}_D$	6	1	16
$\mathcal{S}^F_{ABC/D}$	-	1	1
Total	**18**	**4**	**28**

as constraints in onboard controllers. This could guarantee rigidity without the need for dedicated and complex rigidity-maintenance controllers [113], [198] with global knowledge (such as eigenvectors of the rigidity matrix) that are currently required to ensure rigidity. Beyond its application in formation design, it also has practical relevance in the analysis of singularities appearing in large formations *if* one is able to identify smaller intrinsically rigid subformations within the graph, or when multiple pre-existing formations merge in real time operation.

8.7 Conclusion

This chapter presented a novel perspective on the analysis of singular formation embeddings causing the lose of bearing rigidity. Previous work has focused primarily on the combinatorial nature of rigidity, designing graphs for generic rigidity or rigidity with a specific embedding. Others have studied rigidity using numerical tools, but without a geometric interpretation. The precursor to this work developed a geometric analysis technique for studying the rigidity of formations, but the mathematical complexity limited reasonable applicability of this method to small single-loop formations of 3-4 agents. This chapter extended this geometric analysis by proposing a set-based contraction of robot and inter-robot measurement constraints, which are use to find sufficient conditions for singularity-free operations. This may be performed as an analysis of existing formation

graphs, or for the design of a formation graph with known singularities.

While this method of graph analysis is demonstrated to provide a comprehensive analysis of singularities for graphs that can be fully decomposed into a set of rigid subcomponents with rigid inter-connections, it may be applied to any rigid graph with the the caveat that not all singularities are necessarily identified. There are opportunities to improve the scope of this analysis, primarily through the formal analysis of the $\mathcal{F}_{\mathbb{IOO}}$ and $\mathcal{F}_{\mathbb{OOO}}$ singularities, and through the analysis of multiple interconnected mutually rigidifying loops (as shown in Fig. 8.11). This analysis is furthermore only valid on the $\mathbb{R}^3 \times \mathbb{S}^1$ and sub-manifolds, thus for formations in $SE(3)$ (e.g. omnidirectional UAVs, satellites, and camera networks) this work would need to be extended. For formations evolving on sub-manifolds of $\mathbb{R}^3 \times \mathbb{S}^1$such as $SE(2)$ (e.g. wheeled mobile robots and surface boats) the results of this work are directly applicable, one simply has to restrict the analysis to singularities occurring on the horizontal plane.

To our knowledge it is the first truly systematic attempt to characterize the geometric conditions resulting in such singularities. It avoids the necessity of an ad-hoc mathematical analysis of each individual formation, instead the geometric conditions for many singularities can simply be looked up from tables presented in this chapter. This allows for a greater understanding of the behaviour of formations near singular conditions, and future works are envisioned to attempt to make use of this knowledge in an applied setting.

PART IV

Conclusion and Supplementary Material

Abstract

This part concludes the manuscript with the conclusion, appendices, and bibliography. The conclusion recalls the primary achievements of this work, discusses them within the context of state of the art research in the field, and proposes extensions of this work meriting further study.

There are several appendices included at the end the manuscript, giving details primarily on the experiments. Appendix 1 presents the experimental platform including the hardware and software used during experiments. Appendix 2 discusses the extraction of bearings, first considering the emulated bearings from motion capture, and then the extraction of bearings from onboard cameras. Appendix 3 provides an example of the analysis of the singularities for a simple mechanical structure in order to support the readers understanding of the application of screw theory.

List of Chapters

CHAPTER 9

CONCLUSION

THIS chapter concludes the main body of the thesis, summarizing the contents and contributions presented therein. Our evaluation of how the work falls within the current state of the art is presented, and we discuss the limitations and potential future extensions of this work.

9.1 Summary

This manuscript presents a brief review of general UAV history then presents some typical modelling, control, and sensing systems that are used on quadrotor-style multirotor UAVs. An overview of multi-robot systems is presented with an emphasis of formation control. Decentralized formation controllers using locally available measurements are presented in more detail. Then the manuscript goes on to developing it two main contributions: dynamic formation control and an analysis of formation singularities, discussed hereafter.

9.1.1 Bearing Formation Controllers

This thesis addresses the topic of decentralized bearing formation control, for which the goal is for a group of UAVs to maintain a given spatial geometry using only onboard bearing measurements and limited communication. We begin by proposing a bearing formation controller based on second-order visual servoing (SOVS) by developing the interaction

model relating the bearing feature acceleration to the velocity and acceleration of the observing and observed UAVs. Feedback linearization is used to develop an SOVS controller acting on the acceleration of the observing UAV in order to drive the formation to the given geometry. Additional control features such as actuator saturation, numerical singularities when inverting the interaction matrix, and collective steering by teleoperation are considered. It is found that this controller performs quite well, particularly in scenarios requiring dynamic formation manoeuvring.

Because of the cascading architecture of the quadrotor dynamic model and resulting controllers, it was thought that better (or at least more reactive) control may be achieved by relating the bearing features to the attitude kinematics of the quadrotor. To this end, non-linear model predictive control (MPC) is used to stabilise and steer the formations. Beyond the increased reactivity of the formation, MPC also permits the easy integration of operational constraints, including altitude limits, FOV constraints, and even obstacle avoidance and rigidity maintenance.

9.1.2 Bearing Formation Singularities

During a review of literature on decentralized formation control, the notion of rigidity frequently is used to guarantee that the formation geometry can converge to the desired shape. Many formations are shown to be rigid in most configurations, but in some special configurations (singularities) the rigidity is lost. In this dissertation, we present an overview of the limited existing knowledge of these singularities and also of the much more studied singularities of kinematic mechanisms. We then present a systematic categorization of singularities arising in formation control based on the structure of the formation graph. This is shown to allow the identification of many, if not all singular geometries in much larger formations than hitherto possible. We extend the analysis of existing formations to the design of formations by proving that arbitrarily large formations may be designed for which all singularities lie within a known and bounded set of geometric conditions. This allows us to guarantee that outside of this set of conditions, such a formation is always rigid and therefore the decentralized bearing formation control is achievable.

9.2 Significance of Results

The work contained in this thesis approaches the task of bearing-based formation control from a robotics perspective. The sensor-based formation control is a busy field with constant research and innovation, however most work focuses on single or double integrator dynamics. Those who perform experiments using sensor-based formation control typically control the UAV in velocity using algorithms with mathematically guaranteed convergence. We demonstrated through extensive simulations and real time experiments that using second order visual servoing and model predictive visual servoing can achieve significantly faster dynamic performance, emphasising the importance of considering a more detailed model of the individual robots in the formation control laws.

The work on singularities is to our knowledge the first detailed attempt to characterise degenerate cases of graph rigidity, regardless of the domaine of application. It could therefore be interesting to various inter-disciplinary fields, although we focus here only one bearing formation control. This could be impactful in the design of formations that must work in confined spaces, where collisions may easily occur if formation isn't rigid. Most practical applications of the knowledge of singularities remain to be studied, but this work has partially filled a gap in the field of formation rigidity that has often been remarked upon by the community without a concerted effort to fill.

9.3 Continuity of Research

While this thesis contributed to both the state of the art for bearing formation control and graph rigidity theory, there are many open ends that could be improved upon or extended to further investigation. In this section we will separate future research axes into four main components: 1) the improvement of the developed bearing-based formation controlled 2) the identification and classification of singularities and 3) general axes of interest in the field of decentralized formation control.

With regards to the improvements of the controllers developed herein, future developments must seek either add new functionality to the formation or present better control characteristics. As has already been done using rigidity controllers, potential functions could be added to the SOVS controller to enforce FOV and obstacle avoidance constraints. Furthermore as shown in section 6 the integral term in the SOVS controller

is insufficient to correct for disturbances greater than several percent of the robot weight, and thus a feed-forward disturbance estimated should be envisaged. Concerning the MPC controller, improvements could consist in implementing a more complex model, considering up to the motor torque level as done in [169], and may improve performance with high jerk or snap when manoeuvring. This would likely require a redesign of the experimental setup as it would require the controller to run at high frequency, and the current implementation with ROS communicating with PX4 over serial has too high a latency and a low bandwidth (see appendix A). One could certainly add constraints to the UAVs, indeed certain constraints such as inter-UAV collision avoidance would be trivial to implement but would be highly dependant on a good distance estimate. All of the control constraints however treat the quadrotor model as a purely deterministic process, whereas model and environment uncertainties could be lead to deviations from the predicted path. It may then be interesting to apply tube-MPC [188] where a gaussian process estimates the propagation of uncertainty in the prediction, such that constraints can be enforced in worst-case scenarios. Finally, learning-based control is becomming increasingly used due to it's ability to adapt to systems with uncertain parameters. This may result in improved control when the depth is very difficult to estimate, as well as adapting to issues such as the non-zero time required to redirect the other UAVs in the formation due to a new input, and which are not well accounted for in the prediction. The combination of both MPC and learning-based control methods [190], [191], [217] is particularly promising as MPC can guarentee minimum performances and constraint critical constraint satisfaction while learning-based control may the peak performance.

The second main contribution of this thesis was the identification of singular configurations for generically rigid bearing formations. Our work on the identification of bearing formations was predicated on a graph-based decomposition combined with screw theory, however we show that this analysis is incomplete in the cases where:

- Two rigid subformations are joined by three or more uniquely uni-directional edges
- Three or more rigid subformations are connect without any pair being individually bearing rigid
- When there are multiple closed loops without any intrinsically rigid components

In such cases we may idenitify many singular configurations using the methods presented in this dissertation, but cannot show that the absence of additional singular configurations

have not been identified. This could be a future axis of research making use of more advanced kinematic analysis methods as briefly mentioned in section 7.3. Another aspect of formation singularities which has not been discussed is how to make use of this information. The inclusion of soft constraints penalizing the time spent near singularities could for example allow formations to cross singularities where existing gradient-based methods based on spectral properties [113] would prevent the rigidity becoming zeros and thus the crossing of singularities. Detecting and avoiding local singularities is indeed a direct application of the work done here, however detecting and avoiding subformation singularities in a decentralized manner requires further study. Furthermore when considering the control of rigid formations near singularities and recognizing that singularities consist of an infinitessimally small subspace of $\mathbb{R}^3 \times \mathbb{S}^1$, the question naturally arises as to what is near? There are currently no formal methods do determine the extend of localization uncertainty based on uncertainties in the bearing measurements, formation scale, and robot motion. Approaches based on interval analysis or using machine learning could help answer the question on how much rigidity is required.

Within the broader scope of formation control there are many more possible topics of interest. Relating directly to this work, it would be interesting to explore both the control and the singularity analysis for hybrid formations [218] where agents have access to various combinations of relative positions, bearings, and distances depending on the sensors available on each UAV. From an operational sense, there is much work in making formations more autonomous, rather than decentralized. For example, in our experiments ever agent is given a set of desired bearing measurements, and even when real vision is used we use MOCAP to identify which UAV corresponds to which measurement. For a truly autonomous formation, the UAVs should collaborate together to build the formation without relying on external directions to satisfy a given operational objective, either by building labelled or unlabelled graphs in real time.

APPENDIX A

EXPERIMENTS

SOME chapters in this manuscript presented experimental results and only discuss the aspects relating to the experiments (e.g. the optimization algorithm in chapter 5). This appendix presents the general aspects of the experiments such as the specific hardware and software used.

A.1 The Multirotors

The multirotors are the core of the UAV experimental platform at LS2N, and were designed to provide a easily modified multi-purpose UAV adaptable to the many aerial robotics projects at LS2N.

A.1.1 Hardware

Each quadrotor is built on a 34 cm frame (i.e. between opposing propellers) and weighs approximately 1.025 kg (the actual mass varies by up to $\pm$50 g by depending on the UAV and the battery used). They are equipped with MT2208 1100 kV motors and 12 A electronic speed controllers (ESCs). With 8"$\times$4.5" plastic propellers and a 3-cell LiPo battery, these motors provide up to 4.5 N of thrust each. Each quadrotor is equipped with a set of passive infra-red (IR) reflecting markers in unique patterns so as to be able to be identified by the Qualisys MOCAP system (see Fig. A.1) which uses 8 Miqus IR cameras to capture the pose of rigid bodies in most of the $4 \times 6 \times 4$ m flying arena volume.

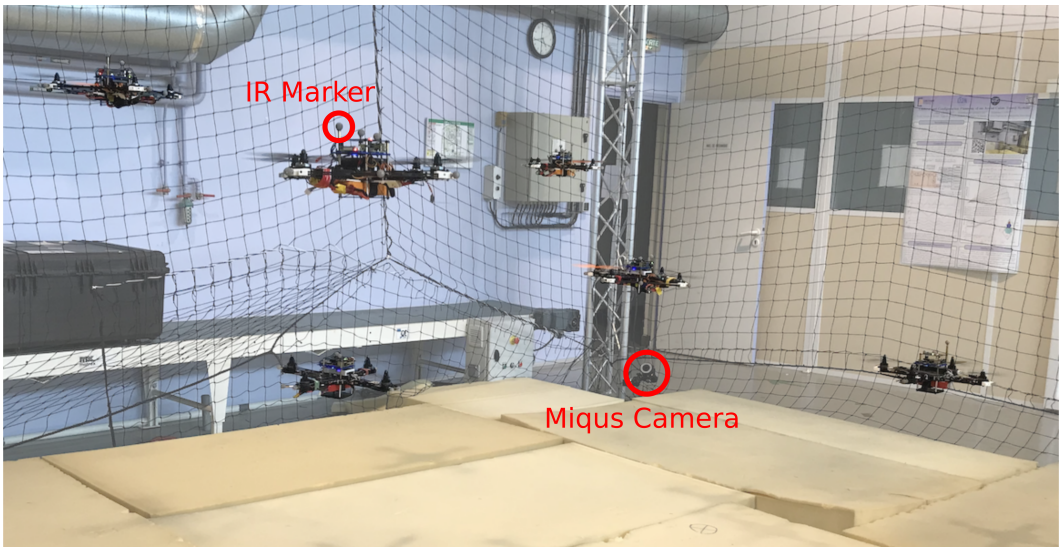


Figure A.1 – Quadrotors flying in the LS2N flight arena

Each quadrotor is equipped with a Pixhawk v2 flight control unit (FCU) which contains the IMU, the actuator interface, and a real-time microprocessor running the PX4 v1.10 flight stack. This performs state estimation and control for the quadrotor at 200 Hz, with the control structure being similar to Fig. 2.8 and able to provide multiple different abstraction levels including manual remote control, and offboard control responding to external setpoints. Offboard control is calculated by a Raspberry Pi 4B (RPi) with 4 GB of RAM running Ubuntu 18, and which communicates with the FCU over a 115 kB/s serial channel using the Mavlink protocol.

A.1.2 Thrust Identification

The controllers used in this thesis require each quadrotor to generate a thrust (in N) along with either a desired attitude or a desired angular velocity. This can be done by using Eq. (2.15) and Eq. (2.11) if the aerodynamic coefficients of the propeller are well identified and the propeller speed tracked in real time. This is generally not done in practice however[1], as it requires ESCs with precise RPM estimation and an identification of each propellers' coefficients, resulting in more expensive and less interchangeable system components. What is generally done instead is to simply control the setpoints of the ESCs as a fraction of its maximum output, and tune attitude controller gains until the flight is satisfactorily stable. With such a setup, in order to implement model-based control a mapping from a force

1. At least for quadrotors. Omnidirectional aerial manipulators often need more precise force control and thus use this method

to a thrust fraction f^d (this mapping can be either linear or quadratic depending on pilot preference).

Rather that build a test bench to identify the relationship between force and thrust fraction, we were able to perform manual flights and use the translational dynamic model of the quadrotor to identify the required mapping. The drone is flown aggressively by a manually operated remote control sending roll, pitch, yaw-rate, and thrust-fraction setpoints. The thrust fraction as well as inertial information are recorded on the flight logger of the FCU at 200 Hz and are used for post-processing. As discussed in chapter 2, the accelerometer measures the specific force on the sensor in the sensor's frame (aligned with the quadrotor frame). This specific force $\mathbf{f}_s$ can be expressed as

$$\mathbf{f}_s = {}^i\mathbf{a}_i - \mathbf{R}_i^T\mathbf{g} \tag{A.1}$$

This can be integrated into the dynamic equation of the quadrotor in $\mathfrak{F}_i$ to give

$$\mathbf{f}_s + \mathbf{R}_i^T\mathbf{g} = \frac{f_i}{m}\mathbf{e}_3 + \mathbf{R}_i^T\mathbf{g} \tag{A.2}$$

and thus the force exerted by the UAV in flight can simply be calculated as

$$f_i = m_i\mathbf{f}_s^T\mathbf{e}_3 \tag{A.3}$$

where m_i is a constant mass measured prior to to flight and $\mathbf{f}_s$ is measured by the accelerometer. A least squares regression is used to fit the thrust fraction command f_i^d to the measured thrust using a linear mapping function

$$f_i = k_1 f_i^d + k_2 V_i + k_3 \tag{A.4}$$

where $k_{1\ldots3}$ are scalar coefficients, and V_i is the measured battery voltage. While considering the battery voltage in the thrust mapping function lead to a lower residual and a more accurate thrust generation, the voltage sensors themselves were sporadically unreliable, and thus for the experiments performed in this thesis the voltage term was removed from Eq. (A.4). The fitting function was applied to a dataset of flights of 60-90 s each, with known masses attached the UAV (so that $1.02 \leq m_i \leq 1.55$ kg for across the experiments), to prevent a heavy grouping of data at the natural hover thrust of the quadrotor. The resulting model

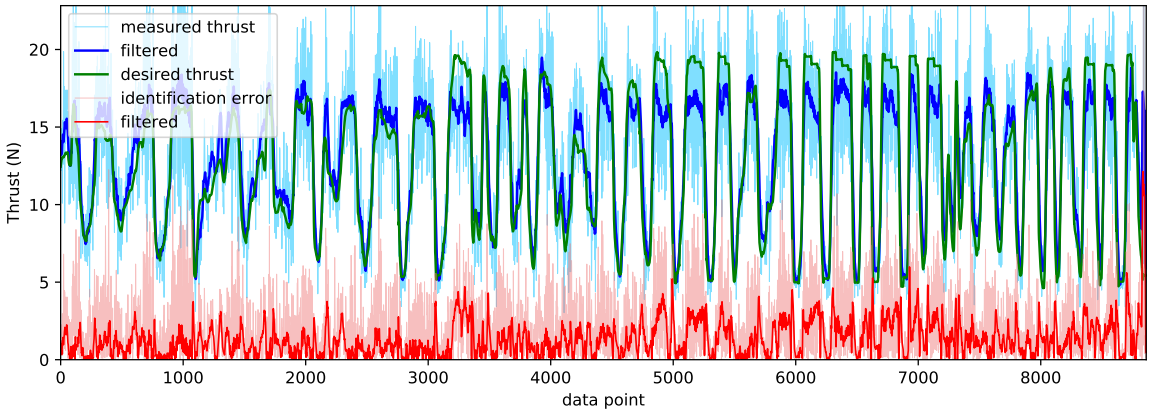


Figure A.2 – Validation data set for thrust parameter identification

was validated on a separate flight as shown in Fig. A.2, and it can be seen that the validation flight overestimates the thrust produced by the quadrotor, particularly towards the end of the flight when the voltage drops. A single UAV was identified in this manner and the resulting model applied to all UAVs with the same hardware components and FCU parameters. While the identification is not perfect, it proved sufficient for the applications of the thesis, as the formation controllers were made robust enough to compensate for small modelling errors. For example in chapter 5, the initial controllers were purely model-based and had different steady state errors between flights (likely in part due to different thrust mapping for different UAVs and batteries), but the disturbance estimate was able to compensate for incorrect thrust modelling by estimating a virtual disturbance force acting on the UAV.

A.2 Information Networking

Flights are performed in the LS2N flight arena, which is equipped with a MOCAP system, which streams the pose of the detected bodies at 100 Hz over a wired local area network (LAN) to the main groundstation computer. The groundstation converts the poses from the MOCAP system to ROS messages, and streams them back over the LAN, to which all UAVs are connected by 5 GHz wifi. The ground station also runs primarily ROS-based programs, written in a mixture of python and C++. These programs include converting joystick inputs to desired motions, generating the desired formation bearings, and synchronizing the start of the experiments[2]. The ground station is also used to launch

2. To start the formation control experiments, each UAV takes off its starting position, indicates to the ground station that it is ready, and hovers until further notice. Once all UAVs are ready, the groundstation sends a message to all the UAVs, allowing them to begin performing formation control

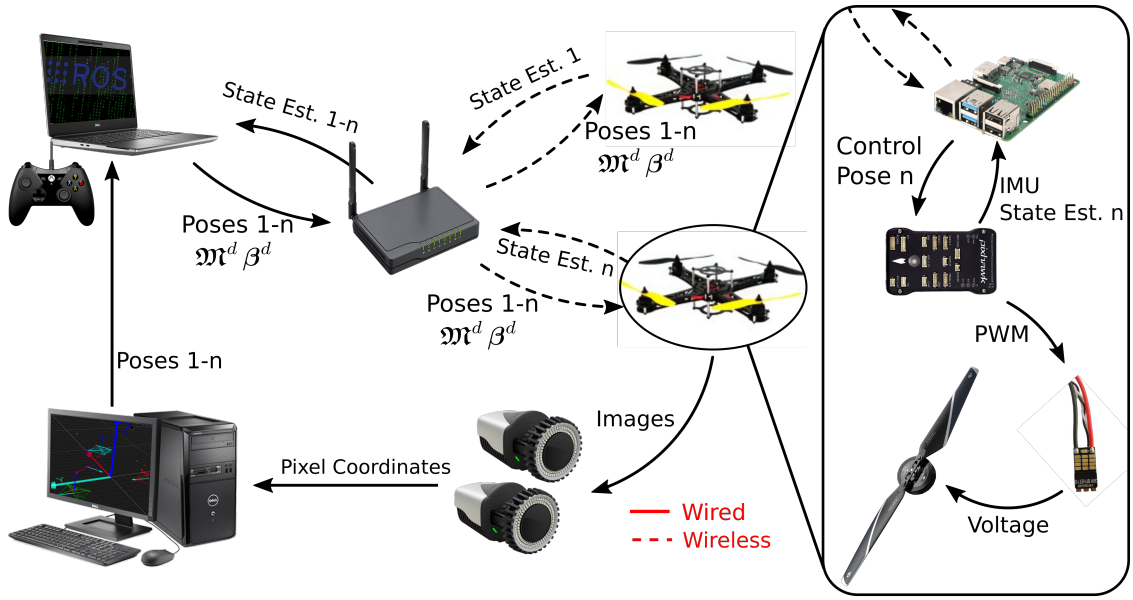


Figure A.3 – Diagram of the experimental setup used in the experiments

programs on the UAVs and for recording data as ROS bags.

Each quadrotor i runs its own controller formation controller onboard, programmed in C++. Bearings $\boldsymbol{\beta}_{ij}$ are calculated using Eq. (2.30), which requires the position of quadrotor j and the position and attitude of quadrotor i[3]. The position of quadrotor j uses the most recently received MOCAP position. Because of time delays and to preserve a decent measure of accuracy in the event that MOCAP is temporarily lost, the position and attitude of quadrotor i is taken from the EKF of $\mathcal{A}_i$ which runs on the FCU. When velocity, acceleration, or angular velocity is needed, these values are also taken from the state estimate provided by the PX4 EKF. All the onboard controllers run at 50 Hz, and although higher-frequency controllers could possibly improve the results (particularly for MPC as it sets angular velocity setpoints) it was found that running the controller faster led to poor performance because of the limited communication bandwidth between the RPi and the Pixhawk.

A.3 Simulations

There is a significant simulation component in this thesis, which we have endeavoured to make as similar to the experiments as possible. Early simulations used Matlab and Simulink, however few of these are presented in the final version of this manuscript. If Matlab or

3. in some experiments where bearings are measured onboard cameras, the MOCAP bearings are still computed to catch false detections or in case the detection algorithm fails to find to UAV for more than 2 s, as discussed in appendix B

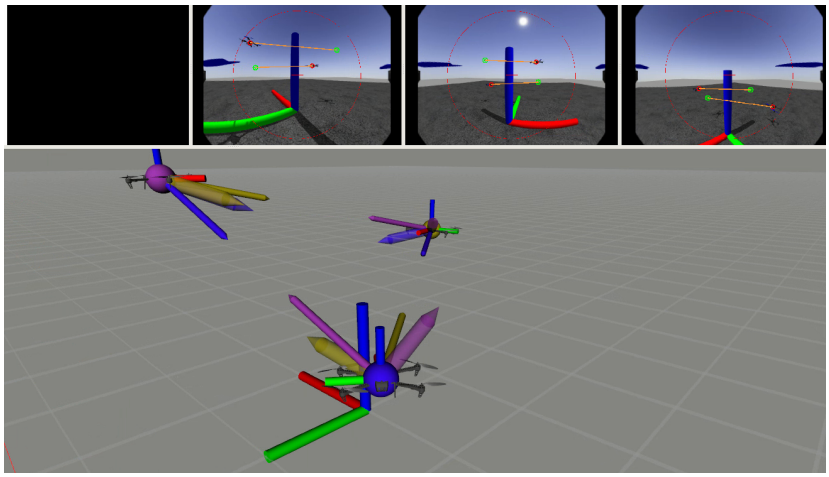


Figure A.4 – Gazebo simulation of a formation. The upper images show the view of the cameras on each UAV rendered by Gazebo, and the lower view shows the RVIZ visualization of the formation, with thick arrows representing the desired bearings and thin arrows representing the measured bearings

simulink is not specified, then the simulations were performed using the RotorS package on Gazebo, using PX4 simulation-in-the-loop (SITL) to mimic the firmware used on the Pixhawks in experiments. This package has been tuned to have similar IMU sensor noise to the UAVs in our experiments, and the thrust coefficients have been empirically identified in the same manner as in section A.1.2. A ROS node has been developed to extract the true pose from Gazebo, and to publish it with comparable noise to that of the MOCAP system, and in the MOCAP message format to enable easy transition from simulations to experiments.

The primary difference between the experiments and the simulations is the lack of communication delays. In experiments, There can be a delay of one control loop between when a MOCAP measurement is taken and when it is used to emulate a bearing measurement. Likewise, estimated states such as velocity and attitude coming from the Pixhawk may be a full control step behind, as the publication speed is limited to 50 Hz to conserve bandwidth on the serial channel. Further delays occur during inter-UAV communication, including delays of multiple control steps occurring between when a state is estimated on $\mathcal{A}_j$ and when it is received and used in the control loop of $\mathcal{A}_i$. This is particularly significant in the SOVS control using the hessian, as it is highly non-linear. Simulations are subject to at most a 5 ms information delay, which is likely the reason for the better steady state error results. A better management of information transfers and real-time handling of the whole formation control pipeline could help to reduce these drawbacks in the experiments, but would require a fairly intensive effort.

APPENDIX B

BEARING DETECTION

BEARING measurements are a core component of the work done in this thesis, however it has only briefly been mentioned that they may be extracted from monocular cameras. This appendix gives more details to that effect, showing how this may be achieved. First a generic pinhole camera model is presented and then the fisheye camera model. The later is more appropriate for formation control, as the wider FOV is critical for persistent tracking, however it comes with image distortion which must be identified. We also discuss how quadrotors may be automatically identified in the images, using deep learning methods with real time performance on drone-scale hardware. Using the MOCAP system to provide a ground truth, we are able to show how how real bearing measurements differ from ideal ones, which is used to evaluate the realism of our experiments.

B.1 Emulated Bearing Accuracy

In many experiments, we use bearings emulated by the MOCAP system. Any given bearing β_{ij} is therefore calculated from Eq. (2.30) given the pose of $\mathcal{A}_i$ and the position of $\mathcal{A}_j$ in $\mathfrak{F}_0$. In this section, we evaluate the static and dynamic bearing uncertainties for these emulated measurements, in order to both evaluate the effective measurement noise used in experiments, and to determine the accuracy of these "ground truth" bearing measurements

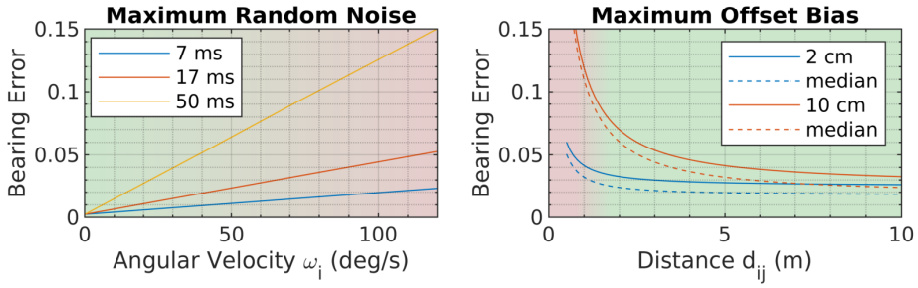


Figure B.1 – The error due to motion capture accuracy and various time delays as a function of the angular velocity of the UAV (left) and the maximum and median errors due to the static frame offset (an angular offset of 1° and various distance offsets) as a function of the measurement distance (right). The background color represents normal (green) and abnormal (red) operating conditions.

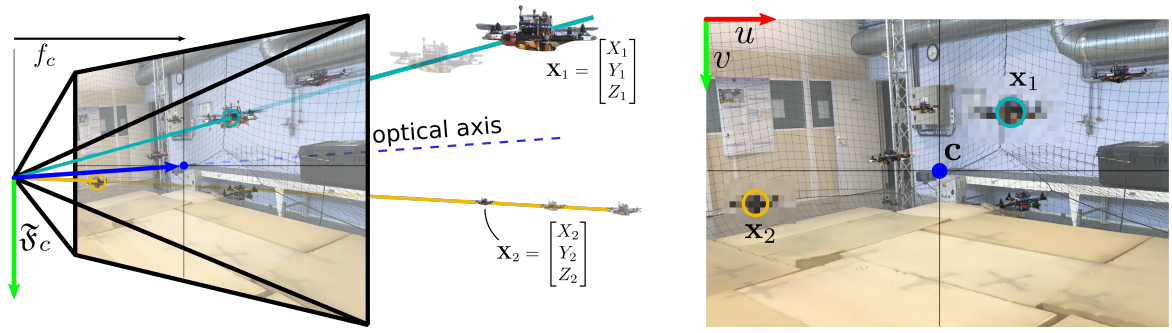
used to evaluate the accuracy of the embedded camera bearing detection later in this chapter.

The MOCAP system has an positioning accuracy of 1 mm and an angular accuracy of 0.1° around each axis. It runs at 100 hz and has a latency of around 7 ms. The positioning of the drone frame in the MOCAP system is only accurate to around 1 cm and 1°, but this is a constant offset throughout the experiment (the true frame $\mathfrak{F}_i$ is defined by the FCU, assumed to be located at the exact geometric centre of the quadrotor). This characterization of the system allows the division of the MOCAP-derived bearing uncertainty into two components:

1. A rapidly changing error due to the angular precision and the unmodelled angular difference due to the angular velocity ω_i of $\mathcal{A}_i$ over the worst case 17 ms between the measurement time and the time the onboard controller receives the measurement. Note that MOCAP positioning precision low enough to ignore.

2. A bias caused by the static difference between the frame of $\mathcal{A}_i$'s flight controller and the frame defined by the MOCAP markers. The angular offset will result in a constant bearing error, whereas the translational offset will be inversely scaled by d_{ij}.

The first source of error is considered random because it difficult to exactly compensate for the angular velocity using gyroscopic measurements due to poor clock synchronization. We can find the maximum regular delay (unless a transmission packet is dropped) but we cannot synchronize the MOCAP and FCU clocks well enough to accurately compensate for this error. Note that the "random" noise during an aggressive flight is only random over a large time scale as it is proportional to the angular velocity. It will therefore naturally damp out as the controller converges, and is large only during transient iterations when a new reference is given. The bearing uncertainty from both sources are plotted in Fig. B.1.



(a) A diagram of the perspective projection of a scene onto an image

(b) The planar image corresponding to the perspective diagram

Figure B.2 – Pinhole camera model

B.2 Monocular Cameras

Monocular cameras are used to record synchronous[1] samples of their perspective view of the world. They are generally composed of a lens which channels light rays coming from the environment through an aperture, forming an planar image on a digital photographic sensor. This sensor measures the light at discrete segments of the sensor called “pixels", encoding the image as a set intensities on blue, green, and red (BGR) channels. The camera model then relates the spatial position of a point relative to the camera to its corresponding coordinate in the planar image.

B.2.1 Pinhole Projection Model

The pinhole camera model is the most basic camera, and works with the assumption that all rays of light travel in a straight line, and intersect at the focal point of the camera lens. Given that a camera observes a point P_j at a relative displacement of ${}^{c}p_j$ expressed in the camera frame $\mathfrak{F}_c$, the pixel coordinates u_j and v_j can be determined by the pinhole camera model

$$\begin{bmatrix} u_j \\ v_j \\ 1 \end{bmatrix} = \lambda_c \begin{bmatrix} f_x & 0 & c_x \\ 0 & fy & c_y \\ 0 & 0 & 1 \end{bmatrix} {}^{c}\mathbf{p}_j \tag{B.1}$$

where c_x and c_y are the principal coordinates of the camera (generally close to the image center) and f_x and f_y are the focal lengths of the camera (which should be very similar, so long as the lens is an acceptable quality and mounted orthogonal to the sensor). As monocular

1. Some asynchronous cameras such as “Event-based cameras" are emerging into robotics domains, but require special treatment beyond the scope of this thesis

cameras cannot measure scale, the is an unknown scaling factor λ_c that is included in the projection model. If one has identified a pixel coordinate $[u, v]$ of interest in an image, one can then of course reconstruct the 3D relative position of the corresponding object such that

$$\begin{bmatrix} p_x \\ p_y \end{bmatrix} = \begin{bmatrix} \frac{p_z(u - c_x)}{f_x} \\ \frac{p_z(v - c_y)}{f_y} \end{bmatrix} \tag{B.2}$$

where the depth p_z of the object is unknown. By arbitrarily setting the depth to $p_z = 1$, the bearing in $\mathfrak{F}_c$ can be expressed as

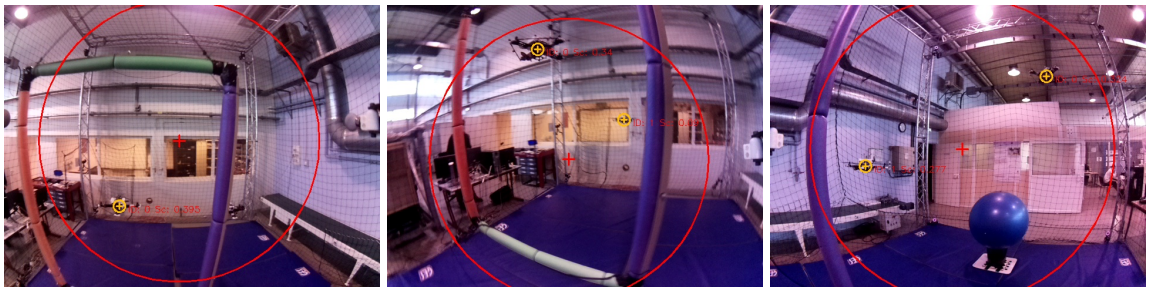
$$\boldsymbol{\beta}_c = \begin{bmatrix} \frac{u - c_x}{f_x} \\ \frac{v - c_y}{f_y} \\ 1 \end{bmatrix} \Bigg/ \left\| \begin{bmatrix} \frac{u - c_x}{f_x} \\ \frac{v - c_y}{f_y} \\ 1 \end{bmatrix} \right\| \tag{B.3}$$

which can be put into the frame $\mathfrak{F}_i$ of the robot as discussed in section B.2.3. First however we will discuss how to calculate the bearing from non-ideal lenses such as fisheye lenses used in our experiments.

B.2.2 Fisheye Distortion and Calibration

Pinhole cameras consider an ideal projection model, however in practice there is distortion caused by the camera lens which is particularly pronounced as the FOV becomes wider. Many distortion models have been proposed and the process for identifying the parameters of the distortion model is implemented in various computer vision libraries. In our case as we use a very wide-angled lens, we use a 4-parameter fisheye distortion model. Without going into much detail, the pinhole camera model is subjected to a polynomial distortion parameterized by the coefficients $d_1 \dots d_4$. Thus in order to reconstruct the bearing from an image coordinate, one needs to know the camera parameters

$$\chi = \begin{bmatrix} f_x & f_y & c_x & c_y & d_1 & d_2 & d_3 & d_4 \end{bmatrix} \tag{B.4}$$



(a) (b) (c)

Figure B.3 – Pictures taken from three onboard cameras. Note the increased distortion (curving of straight lines) towards the edges of each image. Additionally, the red cross-hairs located at the principle axis of each image are different, and the size of the circles (contouring the zone where $||\mathbf{e}_{\beta}|| \leq 0.78$ or $45°$) are different due to variations in focal length and distortion.

There are many camera calibration libraries that can estimate these parameters, in our case we used the openCV v3.4.15 fisheye toolbox[2]. The calibrated distortion parameters may then be used to undistorted any image point so that the projection camera model holds true, and which permits a calculation of the bearings by Eq. (B.3). It is important to realise that the parameters differ significantly between cameras as shown in Fig. B.3 and thus each camera must be individually calibrated, otherwise steady-state bearing errors of $||\mathbf{e}_{\beta}|| \geq 0.15$ per measurement are observed. When a correct calibration is used however, the static bearing measurements were generally found to be $||\mathbf{e}_{\beta}|| \leq 0.02$ (taking MOCAP as a ground truth, with static UAVs separated by ≈2 m).

B.2.3 Frame Transformations

Regardless of the type of camera used, it has been shown that given a point of interest (in this case the center of the target UAV), we are able to project the point onto a unit sphere around the focal point of the camera to recover the bearing in the camera frame $\mathfrak{F}_c$. Because the control is performed in the quadrotor frame $\mathfrak{F}_i$, the bearing $\boldsymbol{\beta}_{ij}$ must by transformed to $\mathfrak{F}_i$. If we let ${}^{i}\mathbf{T}_{c}$ be the homogenous transformation matrix

$$ {}^{i}\mathbf{T}_{c} = \begin{bmatrix} {}^{i}\mathbf{R}_{c} & {}^{i}\mathbf{p}_{c} \\ \mathbf{0} & 1 \end{bmatrix} \tag{B.5} $$

2. https://docs.opencv.org/3.4.15/db/d58/group__calib3d__fisheye.html

The homogenous transformation between the camera frame and the observed quadrotor $\mathcal{A}_j$ can be expressed as

$$^{c}\mathbf{T}_{j} = \begin{bmatrix} ^{c}\mathbf{R}_{j} & \boldsymbol{\beta}_{cj} d_{cj} \\ \mathbf{0} & 1 \end{bmatrix} \tag{B.6}$$

where $\boldsymbol{\beta}_{cj}$ is the bearing of $\mathcal{A}_j$ in $\mathfrak{F}_c$ and $d_{cj} > 0$ is the depth of $\mathcal{A}_j$ with respect to $\mathfrak{F}_c$. The expression of the position of $\mathcal{A}_i$ relative to $\mathfrak{F}_i$ expressed in $\mathfrak{F}_i$ is given by

$$^{i}\mathbf{T}_{j} = {}^{i}\mathbf{T}_{c}\,{}^{c}\mathbf{T}_{j} = \begin{bmatrix} ^{i}\mathbf{R}_{j} & ^{i}\mathbf{p}_{c} + {}^{i}\mathbf{R}_{c}\boldsymbol{\beta}_{cj} d_{cj} \\ \mathbf{0} & 1 \end{bmatrix} \tag{B.7}$$

The bearing $\boldsymbol{\beta}_{ij}$ is simply the normalized position of $\mathfrak{F}_j$ relative to $\mathfrak{F}_i$ which can be extracted directly from the position component of $^{i}\mathbf{T}_{j}$ as

$$\boldsymbol{\beta}_{ij} = \frac{^{i}\mathbf{p}_{c} + {}^{i}\mathbf{R}_{c}\boldsymbol{\beta}_{cj} d_{cj}}{d_{ij}} \tag{B.8}$$

where $d_{ij} = ||^{i}\mathbf{p}_{c} + {}^{i}\mathbf{R}_{c}\boldsymbol{\beta}_{cj} d_{cj}||$. Because the depth is unknown, we can only perform an accurate transformation of the bearing from $\mathfrak{F}_c$ to $\mathfrak{F}_i$ if the origins of $\mathfrak{F}_c$ and $\mathfrak{F}_i$ are coincident ($||^{i}\mathbf{p}_{c}||$ is very small) or if $\mathcal{A}_j$ is very far from $\mathcal{A}_i$ (d_{cj} is very large). As the camera offset is 9 cm on our quadrotors, we can assume that $d_{ij} \approx d_{cj} \gg ||^{i}\mathbf{p}_{c}||$ allowing for the use of the approximate bearing transformation

$$\boldsymbol{\beta}_{ij} = {}^{i}\mathbf{R}_{c}\boldsymbol{\beta}_{cj} \tag{B.9}$$

Having discussed how a $\mathbb{R}^2$ point in an image may be converted to a bearing on the $\mathbb{S}^2$ manifold, we now address the less trivial task of identifying the relevant point in the image.

B.3 Onboard UAV Detection and Bearing Measurement

B.3.1 UAV Detection in Images

Previous works on bearing formation control have made use of motion capture to reconstruct the inter-UAV bearings, but there have also been sever works that use onboard cameras. In [116], [219], coloured spheres are placed on each UAV and circle-fitting techniques are used to detect the spheres after applying color thresholds to the images.

This method necessarily adds mass and additional drag to the UAVs, and additionally may lead to many false positives in uncontrolled environments due to the simple features used. Other more complex features such as the use of fiducial markers [220] may lead to fewer false-positive detections but also require larger objects fixed to the UAVs, while decreasing detection distance and reducing autonomy. Furthermore, marker-based detection methods also require either slow motion or high shutter speed (thus expensive cameras) in order to avoid motion blur caused by the high angular velocities of the camera during aggressive flight. In [221], patterns of active flashing LED lights are used to detect UAVs and would be appropriate for robust operation in many environments including underground and in daylight. These however require that each UAV is equipped with the relevant lights, and also the UAVs become more visible to observers and may potentially be spoofed, thus this is not necessarily appropriate in some scenarios.

While the existing solutions would probably work quite well in our flight arena, the increase in computational power and the development of higher-performance object detection and recognition algorithms led use to try to implement the direct detection of UAVs without markers. Deep learning is a field of machine learning and artificial intelligence where convolutional neural networks (CNNs) are used to identify complex sets of features from images using a set of convolution filters **?**. In our case as the important feature is the center of mass of the observed UAV, we used a modified version of Centernet, a deep learning model design for such a purpose [222]. This was trained using data recovered from several manual flights, and automatically labelled using the MOCAP system. Model inference was performed using an onboard Coral USB Accelerator with image preparation and post-processing performed on the onboard Raspberry Pi. Motion capture was used to remove outlying detections (those with a bearing error greater that 0.3). The total time from image capture to the publication of the bearing messages by the detection node was 0.06-0.08 s. This was slowed down to 5 Hz to reduce computational load, which sometimes lead to issues with the formation controller, particularly when saving images to the SD card during flight. More details about the implementation of the online UAV detection algorithm can be found in [223].

There are three main axes of improvement for the image-based bearing measurements that should be considered moving forward

1. Robustness: The detection worked well when the UAVs were near the center of the

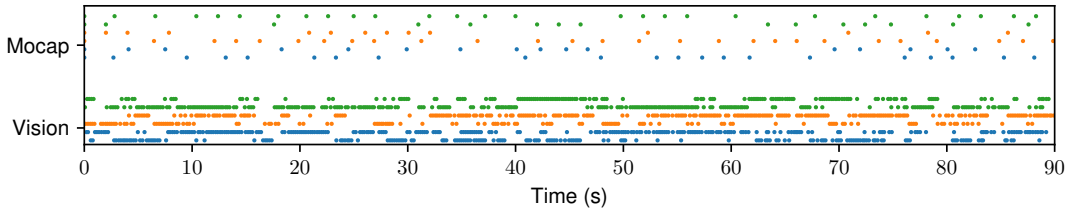
images, however often around the edge, when washed out by light, or against a very dark background, the UAVs were not detected. Part of this may be the fact that the model was trained in 32 and 16 bit precision but that the Coral board performs 8-bit operations and thus the model is highly quantized. An onboard GPU could avoid this issue. There was also limited time to collect training data, and a larger and more diverse data set could improve the results.

2. Tracking: The image detection performs a detection search over the whole image, whereas it would almost certainty be faster and remove false detections if we were to search small portions of the image given the last detected point and the knowledge of the camera motion (extracted from the IMU of the UAV).

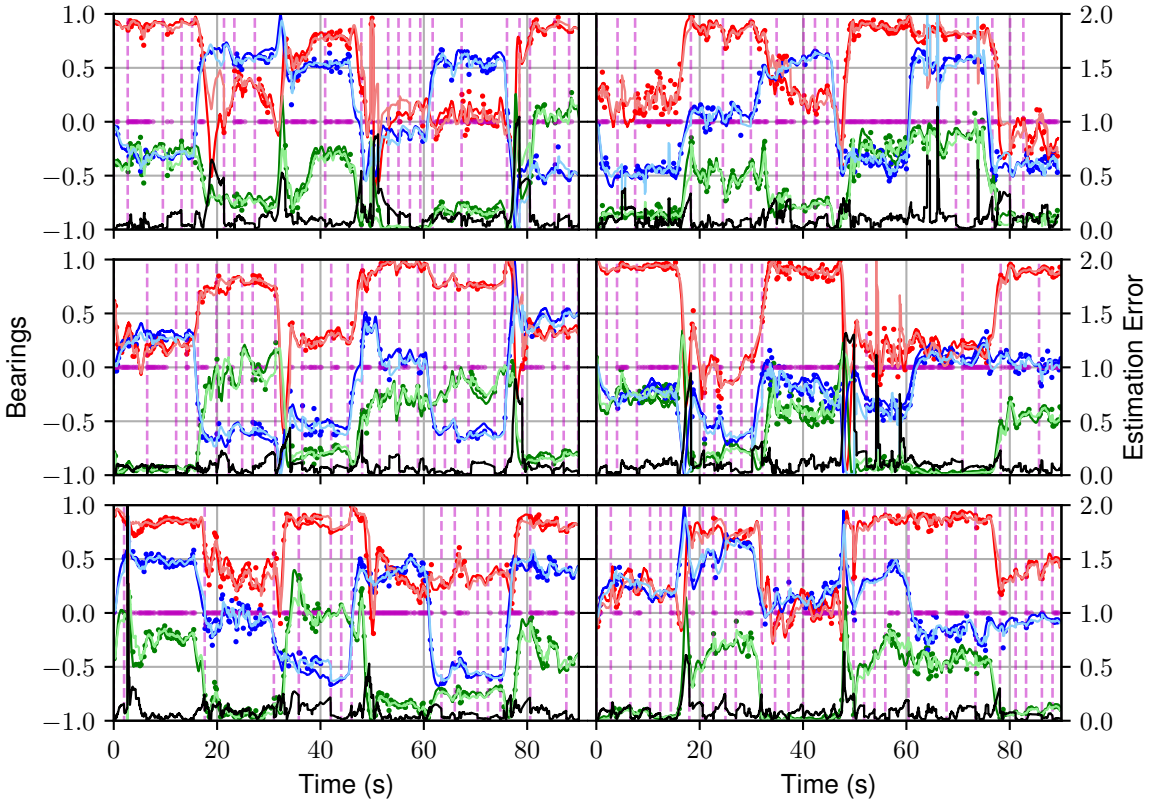
3. Labelling: The UAVs are labelled using MOCAP, as a ground truth is available throughout the whole experiment, even if not used in the control laws. In real situations, determining which UAV in the image corresponds to which position in the formation is not trivial. If an initial labelling is performed, tracking as discussed above can help to keep the initial labelling, even so in the presence of false positive detections such a task is very difficult.

B.3.2 In-Flight Bearing Detection

Having detected the center of the observed UAVs in the image, removed distortion from the image points, and reprojected the resulting bearing from $\mathfrak{F}_c$ to $\mathfrak{F}_i$, all that remains is to use the resulting bearing information in the controller. The bearings could be measured at 10 Hz (or 5 Hz if saving the images), which is must slower than the control loop rate. An extended Kalman filter was then implemented using the bearing interaction model presented in chapter 3 and thus communication of the body-frame velocities are needed between the UAVs. We note that as the bearing measurements evolve on the surface of a sphere and are very non-linear, this is not necessarily the best choice, and an unscented Kalman filter would likely give better results. This is however sufficient to estimate bearings over short gaps in measurements and smooth out measurement noise detections. We also use MOCAP to label which measurement corresponds to which UAV, and to eliminate outlies with a bearing error magnitude greater than 0.3 (although given an accurate initial measurement, there are statistical methods to reduce outliers which could be used in the absence of MOCAP).



(a) The incidence of a bearing measurement obtained from onboard vision (bottom) or from MOCAP (top). The colors represent each UAV, and the two rows for each color correspond to the first and second bearings.



(b) The filtered bearing estimation used by the formation controllers, where the three rows correspond each to a UAV, and the two columns to the two bearing measurements taken by each UAV. The red, green, and blue lines are the components of the bearings drawn on the left axis (dark lines for MOCAP, light lines for the bearing estimator, and dots for camera measurements). The black line (right axis) is the difference between the MOCAP and the estimator. The magenta dots at y=0 indicate the event of capturing a bearing from the image and the vertical dashed lines of using a MOCAP measurement in the estimator.

Figure B.4 – An evaluation of the inter-UAV bearing measurements during an experiment using onboard visual measurements and correcting with a single MOCAP measurements every two seconds where no visual measurement is available.

Because the onboard detection algorithm is somewhat unreliable we cannot always rely on visual measurements, however we wish to make the experiments as realistic as possible. Nonetheless as this thesis is not focused on the robust detection of UAVs in images, we must take some liberties with realism in order to properly test the controllers. As such, if any given bearing is not measured for more than 2 s, a single bearing extracted from MOCAP measurements is supplied to the Kalman filter. For a 90 s experiment (the third experiment of Fig. 5.13) with the formation passing through six different desired shapes and with six directed graph edges, there was a total of 1226 bearing measurements taken. Of these, 1106 were from the onboard vision (90.2%) and 120 were from MOCAP (9.8%). Average out over all bearings, this corresponds to an onboard visual detection at 2.0 Hz, and a MOCAP detection at 0.2 Hz (or an intervention every 5 s). The results of the detection and bearing estimation algorithms are shown in Fig. B.4, and while further work is needed to improve robust and autonomous bearing measurements, for the purpose of testing and developing bearing formation controllers, the bearing detection seems fairly representative of real sensor inputs.

AN EXAMPLE OF SCREW THEORY ANALYSIS

SCREW theory is used in chapter 8 of this manuscript for an extensive analysis of singularities in rigid formations. It is described in chapter 7, nonetheless it a tool that is little used in the multi-robot community, and as such many readers of this thesis may be unfamiliar with it's use. This appendix provides a brief example of the application of screw theory for the kinematic analysis of a simple mechanical structure.

C.1 Wrench Sets of Serial Chains

To demonstrate the concept of reciprocal screws, Fig. C.1 shows an example of a closed-loop kinematic chain composed of two rigid links connected by vertical pins (revolute R joints) to each other and to a rigid ground piece. Dividing the closed loop mechanism into

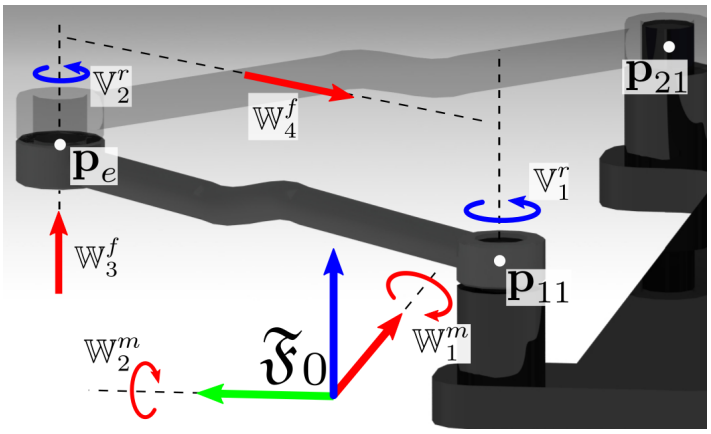


Figure C.1 – Isometric view of an RRR parallel linkage with the twists (in blue) and wrenches (in red) of Limb 1 being labelled.

two serial chains (or "limbs"), the twist set of the first limb (shown in solid color in Fig. C.1) is

$$\mathcal{T}_{\text{RR}} = \left\{ \mathbb{w}_1^{\text{r}}(\mathbf{z}_0, \mathbf{p}_1),\ \mathbb{w}_2^{\text{r}}(\mathbf{z}_0, \mathbf{p}_2) \right\} \tag{C.1}$$

where $\mathbf{p}_1$ and $\mathbf{p}_2$ are the positions of the two R joints in $\mathfrak{F}_0$. The rank of the twist set can be determined by evaluating the rank of a matrix where every column is a twist

$$\mathcal{T}_{\text{RR}} = \begin{bmatrix} \mathbb{w}_1^{\text{r}} & \mathbb{w}_2^{\text{r}} \end{bmatrix} = \begin{bmatrix} \mathbf{z}_0 & \mathbf{z}_0 \\ (\mathbf{z}_0 \times \mathbf{p}_1) & (\mathbf{z}_0 \times \mathbf{p}_2) \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 1 \\ -p_{1y} & -p_{2y} \\ p_{1x} & p_{2x} \\ 0 & 0 \end{bmatrix} \tag{C.2}$$

where p_{ix} and p_{iy} are the x and y components of $\mathbf{p}_i$. It can clearly be seen that as long as the two points are not co-linear along the $\mathbf{z}_0$ axis, than $\text{rank}\left(\mathcal{T}_{RR}\right) = 2$. The degeneration of the serial chain twist set (which adds an additional constraining wrench) is in fact a type of singularity but is not relevant in the context of this paper.

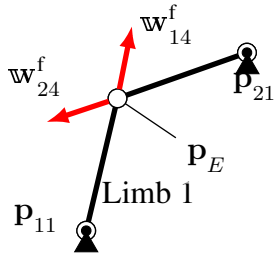
Applying the reciprocity conditions Eq. (7.13) we can find the reciprocal wrench set calculated and expressed in $\mathfrak{F}_0$ as

$$\mathcal{W}_{\text{RR}} = \left\{ \mathbb{w}_1^{\text{m}}(\mathbf{x}_0), \mathbb{w}_2^{\text{m}}(\mathbf{y}_0), \mathbb{w}_3^{\text{f}}(\mathbf{z}_0, \mathbf{p}_2), \mathbb{w}_4^{\text{f}}(\mathbf{p}_{12}, \mathbf{p}_2) \right\} \tag{C.3}$$

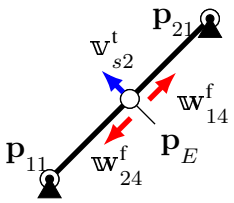
which like the twist set, may be expanded to a matrix

$$\mathcal{W}_{\text{RR}} = \begin{bmatrix} \mathbf{0} & \mathbf{0} & \mathbf{z}_0 & \mathbf{p}_{12} \\ \mathbf{x}_0 & \mathbf{y}_0 & \mathbf{z}_0 \times \mathbf{p}_2 & \mathbf{p}_{12} \times \mathbf{p}_2 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & p_{12x} \\ 0 & 0 & 0 & p_{12y} \\ 0 & 0 & 1 & 0 \\ 1 & 0 & -p_{2y} & 0 \\ 0 & 1 & p_{2x} & 0 \\ 0 & 0 & 0 & \theta \end{bmatrix} \tag{C.4}$$

where $\theta = p_{12x}p_{2y} - p_{12y}p_{2x}$. This wrench set corresponds to two moments spanning the horizontal plane and which keep the revolute joint axes of the the limb aligned with the $\mathbf{z}_0$ direction. It also contains two forces which constraint the motion of the end of the limb,



(a) Top view of an RRR mechanism in a rigid configuration



(b) Top view of an RRR mechanism in a singular configuration

Figure C.2 – A representation of rigid and singular configurations of a planar RRR parallel mechanism composed of two RR limbs. Revolute joints containing a dot are pinned to the ground link.

which can neither move out of the horizontal plane nor compress along its length. These wrenches are drawn in red in Fig. C.1.

For fully constrained mechanisms, the rank of $\mathcal{W}$ is 6, thus the reciprocal twist set $\mathcal{T}$ is empty, meaning that there is no possible motion of the mechanism, i.e. it is rigid. Singularity analysis deals with finding particular configurations of the closed-loop mechanism where rank$\left(\mathcal{W}\right) < 6$, thus leaving the system under-constrained and locally allowing for $6 -$ rank$\left(\mathcal{W}\right)$ singular twists (reciprocal to $\mathcal{W}$).

C.2 Singularities of Closed-Loop Mechanisms

Returning to the example shown in Fig. C.2, two serial RR chains (or limbs) as shown in Fig. C.2a are connected in parallel, forming the closed-loop RRR mechanisms in Fig. C.2(b-c). The first revolute joint of limb i is fixed to a grounded position $\mathbf{p}_{i1}$ (denoted by a dot in the figures), and the second revolute joints are joined together at a common point E at position $\mathbf{p}_E$. As explained before, each limb $i \in 1, 2$ exerts a set of wrenches $\mathcal{W}_{RRi}$, shown in Eq. (C.3), constraining point E resulting in the closed loop wrench set

$$\mathcal{W}_{\text{RRR}} = \mathcal{W}_{\text{RR1}} \cup \mathcal{W}_{\text{RR2}} \tag{C.5}$$

that constrains possible twists of E in $SE(3)$. Note that while this planar mechanism moves on the $SE(2)$ manifold, a wrench set in $SE(3)$ is required to constrain its motion to the $SE(2)$ manifold. The first three wrenches ($\$^{\text{m}}_1$, $\$^{\text{m}}_2$, and $\$^{\text{f}}_3$ from Eq. (C.3)) in $\mathcal{W}_{RRi}$ are identical for both limbs, and the duplicate columns can be eliminated when representing

$\mathcal{W}_{\text{RRR}}$ as the matrix

$$\mathcal{W}_{\text{RRR}} = \begin{bmatrix} 0 & 0 & 0 & p_{Ex} - p_{11x} & p_{Ex} - p_{21x} \\ 0 & 0 & 0 & p_{Ey} - p_{11y} & p_{Ey} - p_{21y} \\ 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & -p_{2y} & 0 & 0 \\ 0 & 1 & p_{2x} & 0 & 0 \\ 0 & 0 & 0 & \theta_1 & \theta_2 \end{bmatrix} \tag{C.6}$$

where $\theta_i = p_{i1x}p_{Ey} - p_{i1y}p_{Ex}$.

The reader will remark that $\mathcal{W}_{RRR}$ can have at most a rank of 5 in a generic configuration (Fig. C.2b). This corresponds to a perpetual singular twist $\mathbb{w}^{\text{r}}_{s1}(\mathbf{z}_0, \mathbf{p}_e)$ corresponding to a rotation of point E about the $\mathbf{z}_0$ axis, as we have considered for simplicity the case where limbs 1 and 2 have superposed revolute joints at their ends. A singular configuration of $\mathcal{W}_{\text{RRR}}$ (Fig. C.2c) occurs when the last two columns ($\mathbb{w}^{\text{f}}_{14}$ and $\mathbb{w}^{\text{f}}_{24}$) become linearly dependant: then rank $\left(\mathcal{W}_{\text{RRR}}\right) = 4$. When all revolute joint centers are aligned, like in Fig. C.2c, then a singular twist $\mathbb{w}^{\text{t}}_{s2}((\mathbf{p}_E - \mathbf{p}_{11}) \times \mathbf{z}_0, \mathbf{p}_E)$ appears, which is reciprocal to all wrenches in $\mathcal{W}_{\text{RRR}}$ and consists of a translation of the end effector in the horizontal plane and orthogonal to the line of the rigid links. This singular twist is an infinitesimal motion that is not constrained by any forces, and can be seen in practice at : https://box.ec-nantes.fr/index.php/s/eLpD5iQLTxR9psE.

BIBLIOGRAPHY

[1] G.-z. Yang, J. Bellingham, P. E. Dupont, P. Fischer, L. Floridi, R. Full, N. Jacobstein, V. Kumar, M. McNutt, R. Merrifield, B. J. Nelson, B. Scassellati, M. Taddeo, R. Taylor, M. Veloso, Z. L. Wang, and R. Wood, "The grand challenges of Science Robotics", *Science Robotics*, vol. 3, *14*, Jan. 2018.

[2] D. Six, A. Chriette, S. Briot, and P. Martinet, "Dynamic Modeling and Trajectory Tracking Controller of a Novel Flying Parallel Robot", *in IFAC-PapersOnLine*, vol. 50, 2017, pp. 2241–2246.

[3] D. Six, S. Briot, A. Chriette, and P. Martinet, "The Kinematics, Dynamics and Control of a Flying Parallel Robot With Three Quadrotors", *IEEE Robotics and Automation Letters*, vol. 3, *1*, pp. 559–566, 2018.

[4] J. Erskine, A. Chriette, and S. Caro, "Wrench Analysis of Cable-Suspended Parallel Robots Actuated by Quadrotors UAVs", *ASME Journal of Mechanisms and Robotics*, vol. 11, *2*, p. 020 909, 2019.

[5] ——, "Control and Configuration Planning of an Aerial Cable Towed System", *in Proceedings of the 2019 IEEE International Conference on Robotics and Automation*, Montreal, 2019.

[6] Z. Hou, "Modeling and formation controller design for multi-quadrotor systems with leader-follower configuratio", PhD thesis, Université de Technologie de Compiègne, 2016.

[7] F. Schiano, "Bearing-based Localization and Control for Multiple Quadrotor UAVs", PhD thesis, Université de Rennes, 2018.

[8] D. Falanga, K. Kleber, and D. Scaramuzza, "Dynamic obstacle avoidance for quadrotors with event cameras", *Science Robotics*, vol. 5, *40*, 2020.

[9] J. Erskine, A. Chriette, and S. Caro, "Wrench Capability Analysis of Aerial Cable-Towed Systems", *in Proceedings of the ASME 2018 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference IDETC/CIE 2018*, Quebec City, 2018, pp. 1–10.

[10] Z. Li, H. Tnunay, S. Zhao, W. Meng, S. Q. Xie, and Z. Ding, "Bearing-Only Formation Control With Prespecified Convergence Time", *IEEE Transactions on Cybernetics*, pp. 1–10, 2020.

[11] D. Six, "Conception et commande de robots parallèles volants", PhD thesis, L'Ecole Centrale de Nantes, 2018.

[12] S. Liu, J. Erskine, A. Chriette, and I. Fantoni, "Decentralized Control and Teleoperation of a Multi-UAV Parallel Robot Based on Intrinsic Measurements", *Preprint*, 2021.

[13] D. Six, S. Briot, J. Erskine, and A. Chriette, "Identification of the Propeller Coefficients and Dynamic Parameters of a Hovering Quadrotor from Flight Data", *IEEE Robotics and Automation Letters*, vol. 5, *2*, pp. 1063–1070, 2020.

[14] R. V. V. Petrescu, R. Aversa, B. Akash, R. Bucinell, J. Corchado, F. Berto, M. Mirsayar, A. Apicella, and F. I. T. Petrescu, "History of Aviation-A Short Review", *Journal of Aircraft and Spacecraft Technology*, vol. 1, *1*, pp. 30–49, 2017.

[15] M. Borri, S. Calabró, and C. Cardani, "Enrico Forlanini's contribution to aviation", *in 43rd European Rotorcraft Forum*, vol. 2, Milan, 2017, pp. 1414–1421.

[16] I. Nicolin and B. A. Nicolin, "The fly-by-wire system", *INCAS Bulletin*, vol. 11, *4*, pp. 217–222, 2019.

[17] K. Sukel, "Come fly with me", *Mechanical Engineering*, pp. 38–43, 2020.

[18] G. Pan and M. S. Alouini, "Flying Car Transportation System: Advances, Techniques, and Challenges", *IEEE Access*, vol. 9, pp. 24 586–24 603, 2021.

[19] L. R. Newcome, *Unmanned Aviation: A Brief History of Unmanned Aerial Vehicles*. American Institude of Aeronautics and Astonautics, Inc., 2004.

[20] J. M. Dunfield and M. Tarbouchi, "Autonomous Hoverable Flying Robot", Royal Military College of Canada, Tech. Rep., 2000, pp. 1–6.

[21] E. Altuğ, J. P. Ostrowski, and R. Mahony, "Control of a quadrotor helicopter using dual camera visual feedback", *in proceedings of the IEEE International Conference on Robotics and Automation*, Washington D.C., 2002, pp. 72–77.

[22] L. A. Young, E. W. Aiken, J. L. Johnson, R. Demblewski, J. Andrews, and J. Klem, "New concepts and perspectives on micro-rotorcraft and small autonomous rotary-wing vehicles", *in proceedings of the 20th AIAA Applied Aerodynamics Conference*, U. Government, Ed., St. Louis, MO, 2002.

[23] T. Hamel, R. Mahony, R. Lozano, and J. Ostrowski, "Dynamic Modelling and Configuration Stabilization for an X4-Flyer", *in IFAC Proceedings Volumes (IFAC-PapersOnline)*, vol. 35, Barcelo, 2002.

[24] Y. Li and S. Song, "A survey of control algorithms for quadrotor unmanned helicopter", *in Proceedings of the 5th International Conference on Advanced Computational Intelligence*, IEEE, Ed., Nanjing, China: IEEE, 2012, pp. 365–369.

[25] S. Gupte, P. I. T. Mohandas, and J. M. Conrad, "A survey of quadrotor unmanned aerial vehicles", *in Proceedings of the IEEE SouthEastCon*, IEEE, 2012.

[26] F. Ruggiero, V. Lippiello, and A. Ollero, "Aerial manipulation: A literature review", *IEEE Robotics and Automation Letters*, vol. 3, *3*, pp. 1957–1964, 2018.

[27] A. Ollero, G. Heredia, A. Franchi, G. Antonelli, K. Kondak, A. Sanfeliu, A. Viguria, J. Ramiro Martinez-De Dios, F. Pierri, J. Cortes, A. Santamaria-Navarro, M. A. T. Soto, R. Balachandran, J. Andrade-Cetto, and A. Rodriguez, "The AEROARMS Project: Aerial Robots with Advanced Manipulation Capabilities for Inspection and Maintenance", *IEEE Robotics and Automation Magazine*, vol. 25, *4*, pp. 12–23, 2018.

[28] M. Tognon, B. Yüksel, G. Buondonno, and A. Franchi, "Dynamic Decentralized Control for Protocentric Aerial Manipulators", LAAS, Toulouse, France, Tech. Rep., 2016.

[29] P. Chermprayong, K. Zhang, F. Xiao, and M. Kovac, "An Integrated Delta Manipulator for Aerial Repair: A New Aerial Robotic System", *IEEE Robotics and Automation Magazine*, vol. 26, *March*, pp. 54–66, 2019.

[30] S. Kim, S. Choi, and H. J. Kim, "Aerial manipulation using a quadrotor with a two DOF robotic arm", *in Proceedings of the IEEE International Conference on Intelligent Robots and Systems IROS*, Tokyo, 2013, pp. 4990–4995.

[31] B. Yüksel, S. Mahboubi, C. Secchi, H. H. Bülthoff, and A. Franchi, "Design , Identification and Experimental Testing of a Light-weight Flexible-joint Arm for Aerial Physical Interaction", *in Proceedings of the IEEE Internation Conference on Robotics and Automation ICRA*, Seattle, 2015.

[32] M. Tognon, C. Gabellieri, L. Pallottino, and A. Franchi, "Aerial Co-Manipulation with Cables: The Role of Internal Force for Equilibria, Stability, and Passivity", *IEEE Robotics and Automation Letters*, vol. 3, *3*, pp. 2577–2583, 2018.

[33] A. Tagliabue, M. Kamel, R. Siegwart, and J. Nieto, "Robust Collaborative Object Transportation Using Multiple MAVs", *International Journal of Robotics Research*, 2016.

[34] S. Thapa, H. Bail, and J. A. Acosta, "Cooperative aerial load transport with attitude stabilization", *in Proceedings of the American Control Conference*, vol. 2019-July, Philadelphia, PA, 2019, pp. 2245–2250.

[35] D. Bicego, "Design and Control of Multi-Directional Thrust Multi-Rotor Aerial Vehicles with applications to Aerial Physical Interaction Tasks", PhD thesis, L'Institut National des Sciences Appliquées de Toulouse, 2019.

[36] M. Tognon and A. Franchi, "Omnidirectional Aerial Vehicles with Unidirectional Thrusters : Analysis , Optimal Design, and Motion Control", *IEEE Robotics and Automation Letters*, vol. 3, *3*, pp. 2277–2282, 2018.

[37] D. Brescianini and R. D'Andrea, "An omni-directional multirotor vehicle", *Mechatronics*, vol. 55, pp. 76–93, 2018.

[38] V. Duggal, M. Sukhwani, K. Bipin, G. S. Reddy, and K. M. Krishna, "Plantation monitoring and yield estimation using autonomous quadcopter for precision agriculture", *in Proceedings of the IEEE International Conference on Robotics and Automation ICRA*, Stockholm, Sweden, 2016, pp. 5121–5127.

[39] R. Beard, "Quadrotor Dynamics and Control Rev 0.1", Tech. Rep., 2008, pp. 1–48.

[40] S. P. Yeong and S. S. Dol, "Aerodynamic optimization of micro aerial vehicle", *Journal of Applied Fluid Mechanics*, vol. 9, *5*, pp. 2111–2121, 2016.

[41] D. D. C. Bernard, F. Riccardi, M. Giurato, and M. Lovera, "A dynamic analysis of ground effect for a quadrotor platform", *in IFAC-PapersOnLine*, vol. 50, Elsevier B.V., 2017, pp. 10 311–10 316.

[42] R. Mahony, V. Kumar, and P. Corke, "Multirotor aerial vehicles: Modeling, estimation, and control of quadrotor", *IEEE Robotics and Automation Magazine*, vol. 19, *3*, pp. 20–32, 2012.

[43] A. Chovancová, T. Fico, E. Chovanec, and P. Hubinský, "Mathematical modelling and parameter identification of quadrotor (a survey)", *Procedia Engineering*, vol. 96, pp. 172–181, 2014.

[44] S. Sun, C. C. De Visser, and Q. Chu, "Quadrotor gray-box model identification from high-speed flight data", *Journal of Aircraft*, vol. 56, *2*, pp. 645–661, 2019.

[45] S. Bouabdallah, "Design and Control of Quadrotors With Application To Autonomous Flying", PhD thesis, École Polytechnique Fédérale De Lausanne, 2007.

[46] M. Hamandi, F. Usai, Q. Sablé, N. Staub, A. Franchi, and M. Tognon, "Survey on Aerial Multirotor Design : a Taxonomy Based on Input Allocation", Tech. Rep., 2020.

[47] M. Zhao, T. Anzai, F. Shi, X. Chen, K. Okada, and M. Inaba, "Design, Modeling, and Control of an Aerial Robot DRAGON: A Dual-Rotor-Embedded Multilink Robot With the Ability of Multi-Degree-of-Freedom Aerial Transformation", *IEEE Robotics and Automation Letters*, vol. 3, *2*, pp. 1176–1183, 2018.

[48] H. Mazeh, M. Saied, H. Shraim, and C. Francis, "Fault-Tolerant Control of an Hexarotor Unmanned Aerial Vehicle Applying Outdoor Tests and Experiments", *IFAC-PapersOnLine*, vol. 51, *22*, pp. 312–317, 2018.

[49] E. Dyer, S. Sirouspour, and M. Jafarinasab, "Energy optimal control allocation in a redundantly actuated omnidirectional UAV", *in Proceedings of the IEEE International Conference on Robotics and Automation ICRA*, Montreal, Canada, 2019, pp. 5316–5322.

[50] T. A. Johansen and T. I. Fossen, "Control allocation - A survey", *Automatica*, vol. 49, *5*, pp. 1087–1103, 2013.

[51] M. Faessler, A. Franchi, and D. Scaramuzza, "Differential Flatness of Quadrotor Dynamics Subject to Rotor Drag for Accurate Tracking of High-Speed Trajectories", *IEEE Robotics and Automation Letters*, vol. 3, *2*, pp. 620–626, 2018.

[52] L. Bauersfeld, E. Kaufmann, P. Foehn, S. Sun, and D. Scaramuzza, "NeuroBEM: Hybrid Aerodynamic Quadrotor Model", *in Robotics: Science and Systems*, 2021.

[53] D. Mellinger and V. Kumar, "Minimum Snap Trajectory Generation and Control for Quadrotors", *in Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, 2011.

[54] D. Mellinger, M. Shomin, N. Michael, and V. Kumar, "Cooperative Grasping and Transport using Multiple Quadrotors", Tech. Rep., 2012, pp. 545–558.

[55] D. Brescianini, M. Hehn, and R. D'Andrea, "Nonlinear Quadrocopter Attitude Control", ETH Zurich, Tech. Rep. 2, 2013.

[56] M. J. Van Nieuwstadt and R. M. Murray, "Real-time trajectory generation for differentially flat systems", *International Journal of Robust and Nonlinear Control*, vol. 8, *11*, pp. 995–1020, 1998.

[57] J. Of and P. Agents, "Multi-Agent versus Multi-Robot and Other Byzantine Discussions", *Journal of Physical Agents*, vol. 2, *1*, pp. 1–3, 2008.

[58] T. Arai, E. Pagello, and L. E. Parker, "Editorial: Advances in Multi-Robot Systems", *Transactions on Robotics and Automation*, vol. 18, *5*, pp. 655–661, 2002.

[59] Z. Y. Lim, S. G. Ponnambalam, and K. Izui, "Multi-objective hybrid algorithms for layout optimization in multi-robot cellular manufacturing systems", *Knowledge-Based Systems*, vol. 120, pp. 87–98, 2017.

[60] M. Dogar, A. Spielberg, S. Baker, and D. Rus, "Multi-robot grasp planning for sequential assembly operations", *Autonomous Robots*, vol. 43, *3*, pp. 649–664, 2019.

[61] M. V. Aerde and S. Y. Agar, "Capacity, Speed, and Platooning Vehicle Equivalents for Two-Lane Rural Highways", *Transportation Research Record*, vol. 971, pp. 58–67, 1984.

[62] A. Al Alam, A. Gattami, and K. H. Johansson, "An experimental study on the fuel reduction potential of heavy duty vehicle platooning", *in Proceedings of the 13th International Conference on Intelligent Transportation Systems*, IEEE, Ed., Madeira Island, Portugal: IEEE, 2010, pp. 306–311.

[63] P. Kavathekar and Y. Chen, "Vehicle Platooning: A Brief Survey and Categorization", *in Proceedings of the ASME Design Engineering Technical Conference IDETC*, Washington D.C., 2011.

[64] L. Jin, M. Čičić, K. H. Johansson, and S. Amin, "Analysis and design of vehicle platooning operations on mixed-traffic highways", *Transactions on Automatic Control*, vol. preprint, 2021.

[65] A. Khalifa, O. Kermorgant, S. Dominguez, and P. Martinet, "Vehicles Platooning in Urban Environment: Consensus-based Longitudinal Control with Limited Communications Capabilities", *in 15th International Conference on Control, Automation, Robotics and Vision, ICARCV*, Signapour: IEEE, 2018, pp. 809–814.

[66] M. Kneissl, A. K. Madhusudhanan, A. Molin, H. Esen, and S. Hirche, "A Multi-Vehicle Control Framework With Application to Automated Valet Parking", *IEEE Transactions on Intelligent Transportation Systems*, pp. 1–11, 2020.

[67] C. Bergenhem, S. Shladover, and E. Coelingh, "Overview of platooning systems", *in Proceedings of the 19th Intelligent Transport Systems World Congress, ITS*, Vienna, Austira, 2012.

[68] P. Culbertson and M. Schwager, "Decentralized Adaptive Control for Collaborative Manipulation", *in Proceedings of the IEEE International Conference on Robotics and Automation*, Brisbane, Australia, 2018, pp. 278–285.

[69] M. Rubenstein, A. Cabrera, J. Werfel, G. Habibi, J. McLurkin, and R. Nagpal, "Collective Transport of Complex Objects by Simple Robots: Theory and Experiments", *in Proceedings of the International Foundation for Autonomous Agents and Multiagent Systems*, Richland, SC, 2013, pp. 47–54.

[70] J. Alonso-Mora, S. Baker, and D. Rus, "Multi-robot formation control and object transport in dynamic environments via constrained optimization", *International Journal of Robotics Research*, vol. 36, *9*, pp. 1000–1021, 2017.

[71] D. Sanalitro, H. J. Savino, M. Tognon, J. Cortés, and A. Franchi, "Full-Pose Manipulation Control of a Cable-Suspended Load with Multiple UAVs under Uncertainties", *IEEE Robotics and Automation Letters*, vol. 5, *2*, pp. 2185–2191, 2020.

[72] H. N. Nguyen, S. Park, J. Park, and D. Lee, "A Novel Robotic Platform for Aerial Manipulation Using Quadrotors as Rotating Thrust Generators", *IEEE Transactions on Robotics*, vol. 34, *2*, pp. 353–369, 2018.

[73] G. Loianno and V. Kumar, "Cooperative transportation using small quadrotors using monocular vision and inertial sensing", *IEEE Robotics and Automation Letters*, vol. 3, *2*, pp. 680–687, 2018.

[74] N. Staub, M. Mohammadi, D. Bicego, Q. Delamare, H. Yang, D. Prattichizzo, P. R. Giordano, D. Lee, and A. Franchi, "The Tele-MAGMaS: An Aerial-Ground Comanipulator System", *IEEE Robotics and Automation Magazine*, vol. 25, *4*, pp. 66–75, 2018.

[75] N. Lissandrini, C. K. Verginis, P. Roque, A. Cenedese, and D. V. Dimarogonas, "Decentralized nonlinear MPC for robust cooperative manipulation by heterogeneous aerial-ground robots", *in Proceedings of the IEEE International Conference on Intelligent Robots and Systems IROS*, 2020, pp. 1531–1536.

[76] L. Huang, M. Zhou, K. Hao, and E. Hou, "A survey of multi-robot regular and adversarial patrolling", *IEEE/CAA Journal of Automatica Sinica*, vol. 6, *4*, pp. 894–903, 2019.

[77] M. Erdelj, M. Król, and E. Natalizio, "Wireless Sensor Networks and Multi-UAV systems for natural disaster management", *Computer Networks*, vol. 124, pp. 72–86, 2017.

[78] J. P. Queralta, J. Taipalmaa, B. Can Pullinen, V. K. Sarker, T. Nguyen Gia, H. Tenhunen, M. Gabbouj, J. Raitoharju, and T. Westerlund, "Collaborative Multi-Robot Search and Rescue: Planning, Coordination, Perception, and Active Vision", *IEEE Access*, vol. 8, pp. 191 617–191 643, 2020.

[79] K. Vencatasamy, L. Jaulin, and B. Zerr, "Secure a Zone from Intruders with a Group Robots", *in Marine Robotics and Applications*, 1, L. Jaulin, A. Caiti, M. Carreras, V. Creuze, F. Plumet, B. Zerr, and A. Billon-Coat, Eds., Springer, 2018, pp. 101–116.

[80] A. Marino, G. Antonelli, A. P. Aguiar, A. Pascoal, and S. Chiaverini, "A decentralized strategy for multirobot sampling/patrolling: Theory and experiments", *IEEE Transactions on Control Systems Technology*, vol. 23, *1*, pp. 313–322, 2015.

[81] D. Portugal and R. Rocha, "A survey on multi-robot patrolling algorithms", *in Technological Innovation for Sustainability: IFIP Advances in Information and Communication Technology*, C.-M. L.M., Ed., vol. 349, Springer, Berlin, Heidelberg, 2011, pp. 139–146.

[82] A. Franchi, P. Stegagno, and G. Oriolo, "Decentralized multi-robot encirclement of a 3D target with guaranteed collision avoidance", *Autonomous Robots*, vol. 40, *2*, pp. 245–265, 2016.

[83] Y. Yu, Z. Li, X. Wang, and L. Shen, "Bearing-only circumnavigation control of the multi-agent system around a moving target", *IET Control Theory & Applications*, pp. 2747–2757, 2019.

[84] G. Lopez-Nicolas, M. Aranda, and Y. Mezouar, "Adaptive Multirobot Formation Planning to Enclose and Track a Target with Motion and Visibility Constraints", *IEEE Transactions on Robotics*, vol. 36, *1*, pp. 142–156, 2020.

[85] P. Schmuck and M. Chli, "Multi-UAV Collaborative Monocular SLAM", *in Proceedings of the IEEE International Conference on Robotics and Automation*, Singapour: IEEE, 2017.

[86] S. Dong, K. A. I. Xu, Q. Zhou, A. Tagliasacchi, S. Xin, M. Niessner, and B. Chen, "Multi-Robot Collaborative Dense Scene Reconstruction", *ACM Transactions on Graphics*, vol. 38, *4*, 2019.

[87] J. Asbach, "Using an Intelligent UAV Swarm in Natural Disaster Environments", *in ASME 2018 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference IDETC/CIE*, Quebec City, 2018.

[88] V. Delafontaine, F. Schiano, G. Cocco, A. Rusu, and D. Floreano, "Drone-aided Localization in LoRa IoT Networks", *in Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, Paris, 2020.

[89] R. D. Arnold, H. Yamaguchi, and T. Tanaka, "Correction to: Search and rescue with autonomous flying robots through behavior-based cooperative intelligence", *Journal of International Humanitarian Action*, vol. 4, *1*, 2019.

[90] M. Haire, X. Xu, L. Alboul, J. Penders, and H. Zhang, "Ship Hull Inspection Using a Swarm of Autonomous Underwater Robots : A Search Algorithm", *in Proceedings of the IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, Wurzburg, Germany, 2019.

[91] S. Milani and A. Memo, "Impact of drone swarm formations in 3D scene reconstruction", *in Proceedings of the IEEE International Conference on Image Processing, ICIP*, Pheonix, AZ, 2016, pp. 2598–2602.

[92] Z. Li, J. Erskine, S. Caro, and A. Chriette, "Design and Control of a Variable Aerial Cable Towed System", *IEEE Robotics and Automation Letters*, vol. 5, *2*, pp. 636–543, 2019.

[93] Z. Li, X. Song, V. Bégoc, A. Chriette, and I. Fantoni, "Dynamic Modeling and Controller Design of a Novel Aerial Grasping Robot", *in Proceedings of ROMANSY: Dynamic Modeling and Controller Design of a Novel Aerial Grasping Robot*, Springer, 2021.

[94] A. Kushleyev, D. Mellinger, C. Powers, and V. Kumar, "Towards a swarm of agile micro quadrotors", *Autonomous Robots*, vol. 35, *4*, pp. 287–300, 2013.

[95] F. Fabra, J. Wubben, C. T. Calafate, J. C. Cano, and P. Manzoni, "Efficient and coordinated vertical takeoff of UAV swarms", *in Proceedings of the 91st IEEE Vehicular Technology Conference*, Antwerp, Belgium, 2020, pp. 1–5.

[96] W. Honig, J. A. Preiss, T. K. Kumar, G. S. Sukhatme, and N. Ayanian, "Trajectory Planning for Quadrotor Swarms", *IEEE Transactions on Robotics*, vol. 34, *4*, pp. 856–869, 2018.

[97] I. L. Cardenas, "Communicating multi-UAV system for cooperative SLAM-based exploration", PhD thesis, Université de Technologie de Compiègne, 2018.

[98] C. W. Reynolds, "Flocks, Herds, and Schools: A Distributed Behavioral Model", *Computer Graphics*, vol. 21, *4*, 1987.

[99] S.-J. Chung, A. A. Paranjape, P. Dames, S. Shen, and V. Kumar, "A Survey on Aerial Swarm Robotics", *IEEE Transactions on Robotics*, vol. 34, *4*, pp. 837–855, 2018.

[100] R. Olfati-Saber, "Flocking for Multi-Agent Dynamic Systems: Algorithms and Theory", *IEEE Transactions on Automatic Control*, vol. 51, *3*, pp. 401–420, 2006.

[101] O. Saif, I. Fantoni, and A. Zavala-Ro, "Distributed Integral Control of Multiple UAVs : Precise Flocking and Navigation", *IET Control Theory and Applications*, vol. 13, *13*, pp. 2008–2017, 2019.

[102] S. Hauert, S. Leven, M. Varga, F. Ruini, A. Cangelosi, J. C. Zufferey, and D. Floreano, "Reynolds flocking in reality with fixed-wing robots: Communication range vs. maximum turning rate", *in Proceedings of the IEEE International Conference on Intelligent Robots and Systems*, San Francisco, CA: IEEE, 2011, pp. 5015–5020.

[103] T. Vicsek, C. Virágh, A. E. Eiben, G. Vásárhelyi, T. Nepusz, G. Somorjai, C. Virágh, G. Somorjai, T. Nepusz, A. E. Eiben, and T. Vicsek, "Optimized flocking of autonomous drones in confined environments", *Science Robotics*, vol. 3, *20*, 2018.

[104] K. Saulnier, D. Saldana, A. Prorok, G. J. Pappas, and V. Kumar, "Resilient Flocking for Mobile Robot Teams", *IEEE Robotics and Automation Letters*, vol. 2, *2*, pp. 1039–1046, 2017.

[105] H. Zheng, J. Panerati, G. Beltrame, and A. Prorok, "An Adversarial Approach to Private Flocking in Mobile Robot Teams", *IEEE Robotics and Automation Letters*, vol. 5, *2*, pp. 1009–1016, 2019.

[106] E. Soria, F. Schiano, and D. Floreano, "The Influence of Limited Visual Sensing on the Reynolds Flocking Algorithm", *Proceedings - 3rd IEEE International Conference on Robotic Computing, IRC 2019*, pp. 138–145, 2019.

[107] F. Schilling, J. Lecoeur, F. Schiano, and D. Floreano, "Learning Vision-based Flight in Drone Swarms by Imitation", *IEEE Robotics and Automation Letters*, vol. 4, *4*, pp. 4523–4530, 2019.

[108] H. Weimerskirch, J. Martin, Y. Clerquin, P. Alexandre, and S. Jiraskova, "Energy saving in flight formation", *Nature*, vol. 413, pp. 697–698, 2001.

[109] T. Marks, K. Dahlmann, V. Grewe, V. Gollnick, F. Linke, S. Matthes, E. Stumpf, M. Swaid, S. Unterstrasser, H. Yamashita, and C. Zumegen, "Climate impact mitigation potential of formation flight", *Aerospace*, vol. 8, *1*, pp. 1–18, 2021.

[110] B. Alkouz and A. Bouguettaya, "Formation-based Selection of Drone Swarm Services", *in International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services*, Darmstadt Germany: Association for Computing Machinery, 2020.

[111] B. D. Anderson, C. Yu, B. Fidan, and J. Hendrickx, "Rigid Graph Control Architectures for Autonomous Formations", *IEEE Control Systems Magazine*, *December*, pp. 48–63, 2008.

[112] X. Li, Y. Tan, J. Fu, and I. Mareels, "On V-shaped flight formation of bird flocks with visual communication constraints", *in Proceedings of the IEEE International Conference on Control and Automation, ICCA*, Ohrid, Macedonia: IEEE, 2017, pp. 513–518.

[113] F. Schiano and P. R. Giordano, "Bearing rigidity maintenance for formations of quadrotor UAVs", *in Proceedings - IEEE International Conference on Robotics and Automation*, Singapore, 2017, pp. 1467–1474.

[114] S. Zhao and D. Zelazo, "Bearing Rigidity Theory and its Applications for Control and Estimation of Network Systems: Life Beyond Distance Rigidity", *IEEE Control Systems Magazine*, pp. 66–83, 2019.

[115] R. Ozawa and F. Chaumette, "Dynamic Visual Servoing with Image Moments for a Quadrotor Using a Virtual Spring Approach", *in Proceedings of the IEEE Internation Conference on Robotics and Automation ICRA*, Shanghai, 2011.

[116] A. Franchi, C. Masone, V. Grabe, M. Ryll, H. H. Bülthoff, and P. R. Giordano, "Modeling and Control of UAV Bearing Formations with Bilateral High-level Steering", *The International Journal of Robotics Research*, vol. 31, *12*, pp. 1504–1539, 2012.

[117] S. Zhao and D. Zelazo, "Bearing Rigidity and Almost Global Bearing-Only Formation Stabilization", *IEEE Transactions on Automatic Control*, vol. 61, *5*, pp. 1255–1268, 2016.

[118] F. Schiano, A. Franchi, D. Zelazo, and P. Robuffo Giordano, "A Rigidity-Based De-centralized Bearing Formation Controller for Groups of Quadrotor UAVs", *in Proceedings of the IEEE International Conference on Intelligent Robots and Systems*, Daejeon, South Korea, 2016, pp. 5099–5106.

[119] Y. Li, S. Zahran, Y. Zhuang, Z. Gao, Y. Luo, Z. He, L. Pei, R. Chen, and N. El-Sheimy, "IMU/magnetometer/barometer/mass-flow sensor integrated indoor quadrotor UAV localization with robust velocity updates", *Remote Sensing*, vol. 11, *7*, pp. 1–22, 2019.

[120] D. Zelazo, A. Franchi, F. Allgöwer, H. H. Bülthoff, and P. Robuffo Giordano, "Rigidity Maintenance Control for Multi-Robot Systems", *in Robotics: Science and Systems Conference*, Sydney, Australia, 2012, pp. 473–480.

[121] D. Zelazo, P. R. Giordano, and A. Franchi, "Bearing-Only Formation Control Using an SE (2) Rigidity Theory", *in IEEE Conference on Decision and Control*, Osaka, 2015.

[122] J. Turek and D. Shasha, "The many faces of consensus in distributed systems", *Computer*, vol. 25, *6*, pp. 8–17, 1992.

[123] N. E. Leonard and E. Fiorelli, "Virtual leaders, artificial potentials and coordinated control of groups", *in Proceedings of the IEEE Conference on Decision and Control*, Orlando, Florida, 2001, pp. 2968–2973.

[124] F. Chaumette and S. Hutchinson, "Visual servo control. I. Basic approaches", *IEEE Robotics and Automation Magazine*, vol. 13, *4*, pp. 82–90, 2006.

[125] F. Fusco, O. Kermorgant, and P. Martinet, "A Comparison of Visual Servoing from Features Velocity and Acceleration Interaction Models", *in Proceedings of the IEEE International Conference on Intelligent Robots and Systems IROS*, Macau, China, 2019.

[126] S. Zhao, "Affine Formation Maneuver Control of Multiagent Systems", *IEEE Transactions on Automatic Control*, vol. 63, *12*, pp. 4140–4155, 2018.

[127] R. Tron, "Bearing-Based Formation Control with Second-Order Agent Dynamics", *in Proceedings of the IEEE Conference on Decision and Control*, Miami Beach, USA: IEEE, 2019, pp. 446–452.

[128] S. Vandernotte, A. Chriette, P. Martinet, and A. S. Roos, "Dynamic sensor-based control", *in Proceedings of the 14th International Conference on Control, Automation, Robotics and Vision, ICARCV*, Phuket, Thailand, 2016.

[129] E. Malis, "Improving vision-based control using efficient Second-Order Minimization Techniques", *in Proceedings of the 2004 IEEE International Conference on Robotics & Automation*, New Orleans: IEEE, 2004, pp. 1843–1848.

[130] S. R. Buss and J.-S. Kim, "Selectively Damped Least Squares for Inverse Kinematics", *Journal of Graphics Tools*, vol. 10, *3*, pp. 37–49, 2005.

[131] V. C. Klema and A. J. Laub, "The Singular Value Decomposition: Its Computation and Some Applications", *IEEE Transactions on Automatic Control*, vol. 25, *2*, pp. 164–176, 1980.

[132] C. D. Martin and M. A. Porter, "The extraordinary SVD", *American Mathematical Monthly*, vol. 119, *10*, pp. 838–851, 2012.

[133] Y. Nakamura and H. Hanafusa, "Inverse kinematic solutions with singularity robustness for robot manipulator control", *Journal of Dynamic Systems, Measurement and Control, Transactions of the ASME*, vol. 108, *3*, pp. 163–171, 1986.

[134] E. Simetti and G. Casalino, "A Novel Practical Technique to Integrate Inequality Control Objectives and Task Transitions in Priority Based Control", *Journal of Intelligent and Robotic Systems: Theory and Applications*, vol. 84, *1-4*, pp. 877–902, 2016.

[135] F. Fusco, O. Kermorgant, and P. Martinet, "Integrating Features Acceleration in Visual Predictive Control", *IEEE Robotics and Automation Letters*, vol. 5, *4*, pp. 5197–5204, 2020.

[136] J. Erskine, R. Balderas-Hill, I. Fantoni, and A. Chriette, "Model Predictive Control for Dynamic Quadrotor Bearing Formations", *in Proceedings of the IEEE International Conference on Robotics and Automation*, Xi'an, China (virtual), 2021.

[137] S. J. Qin and T. A. Badgwell, "A survey of industrial model predictive control technology", *Control Engineering Practice*, vol. 11, pp. 733–764, 2003.

[138] D. Q. Mayne, J. B. Rawlings, C. V. Rao, and P. O. M. Scokaert, "Constrained model predictive control: Stability and optimality", *Automatica*, vol. 36, pp. 789–814, 2000.

[139] D. Pérez-morales, O. Kermorgant, S. Domínguez-quijada, and P. Martinet, "Multi-Sensor-Based Predictive Control for Autonomous Backward Perpendicular and Diagonal Parking", *in Proceedings of the IEEE International Conference on Intelligent Robots and Systems IROS*, Madrid, Spain, 2018.

[140] T. Keviczky, P. Falcone, F. Borrelli, J. Asgari, and D. Hrovat, "Predictive control approach to autonomous vehicle steering", *in Proceedings of the American Control Conference*, vol. 2006, Minneapolis, USA: IEEE, 2006, pp. 4670–4675.

[141] P. Falcone, F. Borrelli, J. Asgariy, H. E. Tsengy, and D. Hrovat, "A model predictive control approach for combined braking and steering in autonomous vehicles", *in 2007 Mediterranean Conference on Control and Automation, MED*, Athens, Greece, 2007.

[142] P. B. Wieber, "Trajectory free linear model predictive control for stable walking in the presence of strong perturbations", *in Proceedings of the 2006 6th IEEE-RAS International Conference on Humanoid Robots*, Genova, Italy: IEEE, 2006, pp. 137–142.

[143] J. Koenemann, A. D. Prete, Y. Tassa, E. Todorov, and O. Stasse, "Whole-body Model-Predictive Control applied to the HRP-2 Humanoid", *in Proceedings of the IEEE International Conference on Intelligent Robots and Systems IROS*, Hamburg, Germany, 2015.

[144] T. Erez, K. Lowrey, Y. Tassa, V. Kumar, S. Kolev, and E. Todorov, "An integrated system for real-time model predictive control of humanoid robots", *in IEEE-RAS International Conference on Humanoid Robots*, Atlanta, GA: IEEE, 2013, pp. 292–299.

[145] H. Diedam, D. Dimitrov, P. B. Wieber, K. Mombaur, and M. Diehl, "Online walking gait generation with adaptive foot positioning through linear model predictive control", *in 2008 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS*, Nice, France: IEEE, 2008, pp. 1121–1126.

[146] M. Diehl, H. G. Bock, H. Diedam, P.-B. Wieber, M. Diehl, H. G. Bock, H. Diedam, P.-b. W. Fast, and D. Multiple, "Fast Direct Multiple Shooting Algorithms for Optimal Robot Control", *in Fast Motions in Biomechanics and Robotics*, Diehl, Moritz and K. Mombaur, Eds., Springer, Berlin, Heidelberg, 2006, p. 28.

[147] H. G. Bock and K. J. Plitt, "Multiple Shooting Algorithm for Direct Solution of Optimal Control Problems.", *IFAC Proceedings Series*, vol. 17, *2*, pp. 1603–1608, 1985.

[148] M. Diehl, H. G. Bock, H. Diedam, P.-B. Wieber, M. Diehl, H. G. Bock, H. Diedam, P.-b. W. Fast, and D. Multiple, "Fast Direct Multiple Shooting Algorithms for Optimal Robot Control", p. 28, 2009.

[149] X. Du, K. K. K. Htet, and K. K. Tan, "Development of a Genetic-Algorithm-Based Nonlinear Model Predictive Control Scheme on Velocity and Steering of Autonomous Vehicles", *IEEE Transactions on Industrial Electronics*, vol. 63, *11*, pp. 6970–6977, 2016.

[150] S. Gros, M. Zanon, R. Quirynen, A. Bemporad, and M. Diehl, "From linear to nonlinear MPC: bridging the gap via the real-time iteration", *International Journal of Control*, vol. 93, *1*, pp. 62–80, 2020.

[151] B. Houska, H. J. Ferreau, and M. Diehl, "An auto-generated real-time iteration algorithm for nonlinear MPC in the microsecond range", *Automatica*, vol. 47, *10*, pp. 2279–2285, 2011.

[152] R. A. Bartlett, A. Wachter, and L. T. Biegler, "Active set vs. interior point strategies for model predictive control", *Proceedings of the American Control Conference*, vol. 6, *June*, pp. 4229–4233, 2000.

[153] H. Nguyen, M. Kamel, K. Alexis, and R. Siegwart, "Model Predictive Control for Micro Aerial Vehicles: A Survey", *arXiv Preprint*, 2020. arXiv: /arxiv.org/abs/2011.11104 [https:].

[154] K. Alexis, C. Papachristos, G. Nikolakopoulos, and A. Tzes, "Model predictive quadrotor indoor position control", *in Proccedings of the 19th Mediterranean Conference on Control and Automation*, Corfu, Greece, 2011, pp. 1247–1252.

[155] M. Abdolhosseini, Y. M. Zhang, and C. A. Rabbath, "An efficient model predictive control scheme for an unmanned quadrotor helicopter", *Journal of Intelligent and Robotic Systems*, vol. 70, pp. 27–38, 2013.

[156] W. Zhao and T. H. Go, "Quadcopter formation flight control combining MPC and robust feedback linearization", *Journal of the Franklin Institute*, vol. 351, pp. 1335–1355, 2014.

[157] M. Bangura and R. Mahony, "Real-time model predictive control for quadrotors", *IFAC Proceedings Volumes (IFAC-PapersOnline)*, pp. 11 773–11 780, 2014.

[158] T. T. Ribeiro, A. G. S. Conceição, I. Sa, and P. Corke, "Nonlinear Model Predictive Formation Control for Quadcopters", *IFAC-PapersOnLine*, vol. 48, *19*, pp. 39–44, 2015.

[159] M. Neunert, C. De Crousaz, F. Furrer, M. Kamel, F. Farshidian, R. Siegwart, and J. Buchli, "Fast nonlinear Model Predictive Control for unified trajectory optimization and tracking", *in Proceedings of the IEEE International Conference on Robotics and Automation*, Stockholm, 2016.

[160] M. Kamel, M. Burri, and R. Siegwart, "Linear vs Nonlinear MPC for Trajectory Tracking Applied to Rotary Wing Micro Aerial Vehicles", *IFAC-PapersOnLine*, vol. 50, *1*, pp. 3463–3469, 2017.

[161] J. C. Pereira, V. J. Leite, and G. V. Raffo, "Nonlinear Model Predictive Control on SE(3) for Quadrotor Aggressive Maneuvers", *Journal of Intelligent and Robotic Systems: Theory and Applications*, vol. 101, *3*, 2021.

[162] L. Peric, M. Brunner, K. Bodie, M. Tognon, and R. Siegwart, "Direct Force and Pose NMPC with Multiple Interaction Modes for Aerial Push-and-Slide Operations", *in Proceedings of the IEEE International Conference on Robotics and Automation*, Xi'an, China, 2021.

[163] D. Lee, H. Seo, D. Kim, and H. J. Kim, "Aerial Manipulation using Model Predictive Control for Opening a Hinged Door", *in Proceedings of the IEEE International Conference on Robotics and Automation*, Paris, France, 2020.

[164] D. Bicego, J. Mazzetto, R. Carli, M. Farina, and A. Franchi, "Nonlinear Model Predictive Control with Enhanced Actuator Model for Multi-Rotor Aerial Vehicles with Generic Designs", *Journal of Intelligent and Robotic Systems: Theory and Applications*, vol. 100, *3-4*, pp. 1213–1247, 2020.

[165] G. Allibert, E. Courtial, and F. Chaumette, "Predictive control for constrained image-based visual servoing", *IEEE Transactions on Robotics*, vol. 26, *5*, pp. 933–939, 2010.

[166] A. Hajiloo, M. Keshmiri, W. F. Xie, and T. T. Wang, "Robust Online Model Predictive Control for a Constrained Image-Based Visual Servoing", *IEEE Transactions on Industrial Electronics*, vol. 63, *4*, pp. 2242–2250, 2016.

[167] B. Penin, P. R. Giordano, and F. Chaumette, "Vision-Based Reactive Planning for Aggressive Target Tracking while Avoiding Collisions and Occlusions", *IEEE Robotics and Automation Letters*, vol. 3, *4*, pp. 3725–3732, 2018.

[168] D. Falanga, P. Foehn, P. Lu, and D. Scaramuzza, "PAMPC: Perception-Aware Model Predictive Control for Quadrotors", *in Proceedings of the IEEE International Conference on Intelligent Robots and Systems*, Madrid, Spain, 2018.

[169] M. Jacquet and A. Franchi, "Motor and Perception Constrained NMPC for Torque-Controlled Generic Aerial Vehicles", *IEEE Robotics and Automation Letters*, vol. 6, *2*, pp. 518–525, 2021.

[170] M. Greeff, T. D. Barfoot, and A. P. Schoellig, "A perception-aware flatness-based model predictive controller for fast vision-based multirotor flight", *IFAC-PapersOnLine*, vol. 53, *2020*, pp. 9412–9419, 2020.

[171] K. Zhang, Y. Shi, and H. Sheng, "Robust Nonlinear Model Predictive Control Based Visual Servoing of Quadrotor UAVs", *IEEE/ASME Transactions on Mechatronics*, vol. 26, *2*, pp. 700–708, 2021.

[172] A. Richards and J. How, "Decentralized model predictive control of cooperating UAVs", *in Proceedings of the IEEE Conference on Decision and Control*, Atlantis, Bahamas: IEEE, 2004, pp. 4286–4291.

[173] I. K. Erunsal, R. Ventura, and A. Martinoli, “Nonlinear Model Predictive Control for 3D Formation of Multirotor Micro Aerial Vehicles with Relative Sensing in Local Coordinates”, *in Proceedings of the IEEE International Symposium on Multi-Robot and Multi-Agent Systems*, New Brunswick, USA, 2019.

[174] E. Camponogara, D. Jia, B. H. Krogh, and S. Talukdar, “Distributed model predictive control”, *IEEE Control Systems Magazine*, pp. 44–52, 2002.

[175] R. Van Parys and G. Pipeleers, “Distributed MPC for multi-vehicle systems moving in formation”, *Robotics and Autonomous Systems*, vol. 97, pp. 144–152, 2017.

[176] R. Tallamraju, S. Rajappa, M. J. Black, K. Karlapalem, and A. Ahmad, “Decentralized MPC based Obstacle Avoidance for Multi-Robot Target Tracking Scenarios”, *in Proceedings of the IEEE International Symposium on Safety, Security, and Rescue Robotics*, Philadelphia, PA: IEEE, 2018, pp. 1–8.

[177] E. Soria, F. Schiano, and D. Floreano, “Predictive control of aerial swarms in cluttered environments”, *Nature Machine Intelligence*, vol. 3, *6*, pp. 545–554, 2021.

[178] C. E. Luis, M. Vukosavljev, and A. P. Schoellig, “Predictive Control for Multi-Robot Motion Planning”, *IEEE Robotics and Automation Letters*, vol. 5, *2*, pp. 604–611, 2020.

[179] F. Chaumette and S. A. Hutchinson, “Visual Servo Control, Part II: Advanced Approaches”, *IEEE Robotics and Automation Magazine*, vol. 14, *March*, pp. 109–118, 2007.

[180] J. Saez-Pons, L. Alboul, J. Penders, and L. Nomdedeu, “Multi-robot team formation control in the GUARDIANS project”, *Industrial Robot*, vol. 37, *4*, pp. 372–383, 2010.

[181] M. Giftthaler, M. Neunert, M. Stauble, and J. Buchli, “The control toolbox - An open-source C++ library for robotics, optimal and model predictive control”, *in Proceedings of the IEEE International Conference on Simulation, Modeling, and Programming for Autonomous Robots*, Brisbane, Australia, 2018, pp. 123–129.

[182] F. Fusco, “Dynamic Visual Servoing for Fast Robotics Arms”, PhD thesis, Ecole Centrale de Nantes, 2020.

[183] D. Falanga, E. Mueggler, M. Faessler, and D. Scaramuzza, “Aggressive quadrotor flight through narrow gaps with onboard sensing and computing using active vision”, *in Proceedings or the IEEE International Conference on Robotics and Automation*, Singapore: IEEE, 2017, pp. 5774–5781.

[184] B. Houska, H. J. Ferreau, and M. Diehl, “ACADO toolkit - An open-source framework for automatic control and dynamic optimization”, *Optimal Control Applications and Methods*, vol. 32, *3*, pp. 298–312, 2011.

[185] M. Kamel, T. Stastny, K. Alexis, and R. Siegwart, “Model predictive control for trajectory tracking of unmanned aerial vehicles using robot operating system”, *Studies in Computational Intelligence*, vol. 707, pp. 3–39, 2017.

[186] D. Guo and K. K. Leang, “Image-based estimation, planning, and control for high-speed flying through multiple openings”, *International Journal of Robotics Research*, vol. 39, *9*, pp. 1122–1137, 2020.

[187] H. Hu, X. Feng, R. Quirynen, M. E. Villanueva, and B. Houska, "Real-Time Tube MPC Applied to a 10-State Quadrotor Model", *Proceedings of the American Control Conference*, vol. 2018-June, *June*, pp. 3135–3140, 2018.

[188] B. T. Lopez, J. P. Howl, and J. J. E. Slotine, "Dynamic tube MPC for nonlinear systems", *in Proceedings of the American Control Conference*, vol. 2019-July, Philadelphia, PA, 2019, pp. 1655–1662.

[189] G. Torrente, E. Kaufmann, P. Fohn, and D. Scaramuzza, "Data-Driven MPC for Quadrotors", *IEEE Robotics and Automation Letters*, vol. 6, *2*, pp. 3769–3776, 2021.

[190] B. Tearle, K. P. Wabersich, A. Carron, and M. N. Zeilinger, "A predictive safety filter for learning-based racing control", *Preprint*, 2021. arXiv: 2102.11907.

[191] R. Soloperto, J. Kohler, and F. Allgower, "Augmenting MPC schemes with active learning: Intuitive tuning and guaranteed performance", *IEEE Control Systems Letters*, vol. 4, *2*, pp. 713–718, 2020.

[192] A. De Luca, G. Oriolo, and P. Robuffo Giordano, "Feature depth observation for image-based visual servoing: Theory and experiments", *International Journal of Robotics Research*, vol. 27, *10*, pp. 1093–1116, 2008.

[193] S. Zhao, Z. Sun, D. Zelazo, M. H. Trinh, and H.-S. Ahn, "Laman graphs are generically bearing rigid in arbitrary dimensions", *in Proceedings of the IEEE 56th Annual Conference on Decision and Control*, Melbourne, Australia: IEEE, 2017, pp. 3356–3361.

[194] J. Hendrickx, B. D. Anderson, J.-C. Delvenne, and V. D. Blondel, "Directed graphs for the analysis of rigidity and persistence in autonomous agent systems", *International Journal of Robust and Nonlinear Control*, vol. 17, pp. 960–981, 2007.

[195] R. Tron, L. Carlone, F. Dellaert, and K. Daniilidis, "Rigid components identification and rigidity control in bearing-only localization using the graph cycle basis", *in Proceedings of the American Control Conference*, Chicago, USA: American Automatic Control Council, 2015, pp. 3911–3918.

[196] L. Asimow and B. Roth, "The Rigidity of Graphs", *Transactions of the American Mathematical Society*, vol. 245, *November 1978*, pp. 171–190, 1978.

[197] F. Schiano and R. Tron, "The Dynamic Bearing Observability Matrix Nonlinear Observability and Estimation for Multi-Agent Systems", *in Proceedings of the IEEE International Conference on Robotics and Automation ICRA*, Brisbane, Australia, 2018, pp. 3669–3676.

[198] D. Zelazo, A. Franchi, H. H. Bülthoff, and P. Robuffo Giordano, "Decentralized rigidity maintenance control with range measurements for multi-robot systems", *International Journal of Robotics Research*, vol. 34, *1*, pp. 105–128, 2015.

[199] F. Buckens, "On the Singular Configurations of Statically Determinate Structures", *International Journal of Mechanical Science*, vol. 7, pp. 301–314, 1965.

[200] B. Paul and D. Krajcinovic, "Computer analysis of machines with planar motion: Part 1-Kinematics", *Journal of Applied Mechanics, Transactions ASME*, vol. 37, *3*, pp. 703–712, 1970.

[201] C. Gosselin and J. Angeles, "Singularity Analysis of Closed-Loop Kinematic Chains", *IEEE Transcations on Robotics and Automation*, vol. 6, *3*, pp. 281–290, 1990.

[202] D. Zlatanov, R. G. Fenton, and B. Benhabib, "Singularity analysis of mechanisms and robots via a motion-space model of the instantaneous kinematics", *in Proceedings of the IEEE International Conference on Robotics and Automation*, vol. 2, San Diego, California, USA, 1994, pp. 980–985.

[203] E. Marchand, F. Chaumette, and A. Rizzo, "Using the task function approach to avoid robot joint limits and kinematic singularities in visual servoing", *in Proceedings of the IEEE International Conference on Intelligent Robots and Systems*, vol. 3, Osaka, Japan, 1996, pp. 1083–1090.

[204] K. A. O'Neil, Y. C. Chen, and J. Seng, "Removing singularities of resolved motion rate control of mechanisms, including self-motion", *IEEE Transactions on Robotics and Automation*, vol. 13, *5*, pp. 741–751, 1997.

[205] X. Kong and C. Gosselin, *Type synthesis of Parallel Mechanisms*, B. S. Groen, O. Khatib, and F. Groen, Eds. Springer, 2015, vol. 33.

[206] R. S. Ball, *Theory of Screws: A Study in the Dynamics of a Rigid Body*. Dublin, Ireland: Hodges, Foster, and Co., 1876.

[207] P. Long, W. Khalil, and S. Caro, "Kinematic analysis of lower mobility cooperative arms by screw theory", *in Proceedings of the 9th International Conference on Informatics in Control, Automation and Robotics*, Rome, Italy, 2012, pp. 280–285.

[208] S. Briot, P. Martinet, and V. Rosenzveig, "The hidden robot: An efficient concept contributing to the analysis of the controllability of parallel robots in advanced visual servoing techniques", *IEEE Transactions on Robotics*, vol. 31, *6*, pp. 1337–1352, 2015.

[209] S. Briot and P. Robuffo Giordano, "Physical Interpretation of Bearing Rigidity Graphs : Application to Mobility and Singularity Analyses", *ASME Journal of Mechanisms and Robotics*, vol. 11, *3*, pp. 031006–1–10, 2019.

[210] S. Briot, F. Chaumette, and P. Martinet, "Revisiting the determination of the singularity cases in the visual servoing of image points through the concept of hidden robot", *IEEE Transactions on Robotics*, vol. 33, *3*, pp. 536–546, 2017.

[211] M. Slavutin and Y. Reich, "Singularity analysis of some multi-platform mechanisms by decomposition and reciprocality", *Mechanism and Machine Theory*, vol. 146, 2020.

[212] N. Baron, A. Philippides, and N. Rojas, "On the False Positives and False Negatives of the Jacobian Matrix in Kinematically Redundant Parallel Mechanisms", *IEEE Transactions on Robotics*, pp. 1–8, 2020.

[213] M. Slavutin, O. Shai, A. Sheffer, and Y. Reich, "A novel criterion for singularity analysis of parallel mechanisms", *Mechanism and Machine Theory*, vol. 137, pp. 459–475, 2019.

[214] S. White and P. Smyth, "A spectral clustering approach to finding communities in graphs", *Proceedings of the 2005 SIAM International Conference on Data Mining, SDM 2005*, pp. 274–285, 2005.

[215] S. Rajamanickam and E. Boman, "Parallel partitioning with Zoltan: Is hypergraph partitioning worth it", *in Graph Partitioning and Graph Clustering*, D. Bader, H. Meyerhenke, P. Sanders, and D. Wagner, Eds., American Mathematical Society, 2013, pp. 37–52.

[216] S. E. Schaeffer, “Graph clustering”, *Computer Science Review*, vol. 1, pp. 27–64, 2007.

[217] Y. Song and D. Scaramuzza, “Learning High-Level Policies for Model Predictive Control”, *in Proceedings of the IEEE International Conference on Intelligent Robots and Systems*, Las Vegas, NV, 2020.

[218] S.-H. Kwon, Z. Sun, B. D. Anderson, and H.-S. Ahn, “Hybrid rigidity theory with signed constraints and its application to formation shape control in 2-D space”, *in Proceedings of the IEEE Conference on Decision and Control*, Jeju Island, Korea, 2020, pp. 518–523.

[219] R. Tron, J. Thomas, G. Loianno, K. Daniilidis, and V. Kumar, “A distributed optimization framework for localization and formation control”, *IEEE Control Systems Magazine*, vol. 36, *4*, 2016.

[220] M. Kalaitzakis, S. Carroll, A. Ambrosi, C. Whitehead, and N. Vitzilaios, “Experimental Comparison of Fiducial Markers for Pose Estimation”, *in International Conference on Unmanned Aircraft Systems*, Athens, Greece, 2020, pp. 781–789.

[221] V. Walter, N. Staub, A. Franchi, and M. Saska, “UVDAR System for Visual Relative Localization With Application to Leader–Follower Formations of Multirotor UAVs”, *IEEE Robotics and Automation Letters*, vol. 4, *3*, pp. 2637–2644, 2019.

[222] X. Zhou, D. Wang, and P. Krahenbuhl, “Objects as Points”, *in arXiv preprint*, 2019. arXiv: 1904.07850.

[223] K. Anyinam-Boateng, “Master Thesis: Deep Learning-based Automatic UAV Detection using Embedded Fish-eye Cameras”, Ecole Centrale de Nantes, Nantes, France, Tech. Rep., 2021.

FIGURE REFERENCES

[F1] C.-L. Desraie. (1783). Drawing of mongolfier balloon flight - bibliotheque nationale de france. Accessed on 2021-10-12, [Online]. Available: https://commons.wikimedia.org/wiki/File:Montgolfiere_1783.jpg.

[F2] (2013). Reconstruction of the experimental helicopter of enrico forlanini (1877) - museo nazionale della scienza e della tecnologia leonardo da vinci, milano. Accessed on 2021-10-12, [Online]. Available: https://commons.wikimedia.org/wiki/File:Elicottero_sperimentale_Enrico_Forlanini_1877_Museo_scienza_e_tecnologia_Milano.jpg.

[F3] J. Daniels. (1903). Library of congress, prints and photographs division, lc-dig-ppprs-00626. Accessed on 2021-10-12, [Online]. Available: http://www.loc.gov/pictures/resource/ppprs.00626/.

[F4] Zipline. (2021). Zipline drone delivers medical supplies by parachute. Accessed on 2021-10-12, [Online]. Available: https://aviationweek.com/air-transport/aircraft-propulsion/zipline-expands-medical-drone-delivery-africa.

[F5] US Department of Transport. (2019). FAA helicopter flying handbook, chapter 4: helicopter components, sections, and systems. Accessed on 2021-10-12, [Online]. Available: https://www.faa.gov/regulations_policies/handbooks_manuals/aviation/helicopter_flying_handbook/.

[F6] (). Generic quadrotor, (visited on).

[F7] Intel. (2021). Intel drone light shows. Accessed on 2021-10-12, [Online]. Available: https://inteldronelightshows.com/.

[F8] (2015). BBC article "your help needed to monitor starling murmurations". Accessed on 2021-10-12, [Online]. Available: https://www.bbc.co.uk/newsround/34897890.

[F9] Unknown. (). Birds in leader-follower formation. Accessed on 2021-10-12, [Online]. Available: https://www.nicepng.com/maxp/u2w7u2q8u2q8u2w7/.

Titre : Commande Dynamique et Singularités des Formations de Quadrirotors Basées sur des “Bearings”

Mot clés : Quadrirotor, Formations multi-robots, Commande prédictive, Singularitiés, Rigidité

Résumé : Le contrôle des formations basées sur les bearings (direction relative à l’observateur) permettent aux flottes de quadrirotors de se déplacer vers une géométrie désirée, en utilisant des mesures extraites de caméras embarquées. Des travaux antérieurs ont traité les quadrirotors comme des intégrateurs, et donc la formation doit ralentir de manière à compenser les non-linéarités non modélisées. Cette thèse a pour objectif d’atteindre des formations rapides en tenant compte des dynamiques non-linéaires du quadrirotor et des mesures visuelles. Deux contrôleurs sont développés, à savoir un contrôleur basé sur un asservissement visuel dynamique et une commande prédictive, montrant des performances améliorées avec des contraintes réelles.

Toutes les formations basées sur des bearings dépendent d’un degré suffisant de rigidité. Bien que celui-ci puisse être évalué numériquement, la rigidité est une fonction de la position de tous les robots dans la flotte. Ceci étant, les travaux précédents ne pouvaient pas garantir la rigidité pour des formations plus larges que quelques robots. La deuxième contribution de cette thèse est l’évaluation des géométries singulières où une certaine formation rigide devient flexible. Ceci mène à un système de classification basé sur des contractions d’ensembles de contraintes, qui permet d’identifier les géométries singulières pour des grandes formations afin de garantir la rigidité.

Title: Dynamic Control and Singularities of Rigid Bearing-Based Formations of Quadrotors

Keywords: Quadrotors, Bearing formations, Predictive control, Singularities, Rigidity

Abstract: Bearing formation control allows groups of quadrotors to manoeuver in a desired geometry, using only visual measurements extractable from embedded monocular cameras. Prior works have treated quadrotors as single or double integrators, and as a result must operate slowly to compensate for unmodelled non-linearities. This thesis allows for faster bearing formations by developping higher-order controllers, considering the non-linear quadrotor and visual feature dynamics. A dynamic feedback controller based on second-order visual servoing and a model predictive controller are developped and tested in simulation and experiments, showing improved dynamic manoeuvering performance. The later is augmented with constraints such as field of view limitations and obstacle avoidance.

All bearing formation algorithms depend on a sufficient degree of bearing rigidity to guarantee performance. This may be evaluated numerically, but as the rigidity is a function of the formation embedding, previous work could not guarantee rigidity in formations larger than a few robots. The second main contribution of this thesis is the evaluation of bearing rigidity singularities (i.e. embeddings where an otherwise rigid formation becomes flexible) by applying existing geometric analysis methods on an kinematic mechanism which is analoguous to the kinematic constraints imposed by the formation controller and robot models. This is extended to a novel classification system based on a contraction of constraint sets that can determine singular geometries for large formations, allowing for a formulation of a set of guaranteed rigid configurations without an ad-hoc kinematic analysis of individual formations.