Autonomous indoor navigation of a UAV : single image based.

Jose Juan Téllez-Guzmán

Communauté
UNIVERSITÉ Grenoble Alpes

THÈSE

Pour obtenir le grade de

DOCTEUR DE LA COMMUNAUTÉ UNIVERSITÉ GRENOBLE ALPES

Spécialité : AUTOMATIQUE - PRODUCTIQUE

Arrêté ministériel : 25 mai 2016

Présentée par

JOSE JUAN TELLEZ-GUZMAN

Thèse dirigée par **Nicolas MARCHAND**, CNRS
et codirigée par **José-Ernesto GOMEZ BALDERAS**
préparée au sein du **Laboratoire Grenoble Images Parole Signal Automatique**
dans **l'École Doctorale Electronique, Electrotechnique, Automatique, Traitement du Signal (EEATS)**

Navigation autonome d'un drone en intérieur basée sur les images

Autonomous indoor navigation of a UAV: single image based.

Thèse soutenue publiquement le **26 juin 2019**,
devant le jury composé de :

Monsieur NICOLAS MARCHAND
DIRECTEUR DE RECHERCHE, CNRS DELEGATION ALPES, Directeur de thèse
Monsieur PEDRO CASTILLO
CHARGE DE RECHERCHE, CNRS, Co-directeur de thèse
Madame ISABELLE FANTONI
DIRECTEUR DE RECHERCHE, CNRS, Rapporteur
Monsieur CEDRIC DEMONCEAUX
PROFESSEUR, IUT - LE CREUSOT - BOURGOGNE, Rapporteur
Madame MICHELE ROMBAUT
PROFESSEUR, UNIVERSITE GRENOBLE ALPES, Président
Monsieur GUILLAUME ALLIBERT
MAITRE DE CONFERENCES, UNIVERSITE COTE D'AZUR, Examinateur

A dissertation submitted for the degree of Doctor of Philosophy of the Université de Grenoble

❧

Électronique, Électrotechnique, Automatique et Traitement du Signal (EEATS)

Field : Automatique-Productique

Autonomous indoor navigation of a UAV: single image based

By : **José Juan Téllez Guzmán**

Under the direction of Nicolas Marchand, Pedro Casillo, José Ernesto Gomez Balderas **and** Amaury Negre

Doctoral Committe:

President: Michèle ROMBAUT, Professor, Université Grenoble Alpes,

Reviewer: Isabelle FANTONI, Directeur de recherche au CNRS, membre du LS2N, laboratoire, à Centrale Nantes

Reviewer: Cédric DEMONCEAUX, Professor, IUT LE CREUSOT LABORATOIRE Le2i12,

Examiner: Guillaume ALLIBERT, Maître de conferences, Université Côte d'Azur, CNRS, I3S-France.

Date: September 23, 2019

Aknowledgments

First of all, I want to express my gratitude to my advisors, professors Nicolas MARCHAND, Pedro CASTILLO, Jose Ernesto GOMEZ BALDERAS and Amaury NEGRE for their counseling during the development of my Ph.D studies, their kindness and friendship. Thank you for always have your doors open for questions or any other kind of support.

I want to thank the jury members, Isabele FANTONI, Cèdric DEMONCEAUX, Michèle ROMBAUT and Guillaume ALLIBERT for accepting to review this work and for their suggestions to improve the quality of this work.

My gratitude also to the engineering team of GIPSA-lab, especially to Jonathan DUMON, for his support during the experimental development of the project.

I want to thank also to my colleagues and friends in GIPSA, for creating an environment of friendship and for always giving me their support whenever I was in need of it. Special thanks to my mexican friends, Jonatan ALVAREZ MUNOS, Josué COLMENARES VAZQUEZ, Jorge VELAZQUEZ RODRIGUEZ and Nicolas ESPITIA, for the numerous discussions, support and camaraderie during my stay in Grenoble.

My thoughts go out to my mom, my sister and brother, who have always supported me. Thank you from the deep of my heart.

In memorial to my father who will always be in my heart...

Contents

List of Figures

Abstract

This thesis presents the design and practical implementation of a quadrotor indoor navigation system using a vision system whose input data are obtained from environment information acquired by an embed- ded camera placed on quadrotor system. Actually, some used techniques are mono vision, stereo vision, SLAM among others. For our research we propose to work with mono vision system, due the limited payload of a quadrotor system. Properties in image perspective are used to design the embedded vision system which aim is to extract visual information, to fly placed always in the center of a corridor. Camera rotation ma- trix is obtained by means of orthogonal directions extracted by vanishing points, which directions define a common structured environment. Then, to control and to stabilize our quadrotor system, a quaternion bounded control scheme is presented, which stabilize quadrotor's orientation, and also is used to control its heading direction merging visual information.

Quadrotor estimation positions with respect to world reference in y and z-axes are used as input for the bounded position control to pose it at desired position. It should be mentioned that vision strategy is not able to estimate x-axis, thus this axis is controlled manually. In order to corroborate ours results, mathematical model, control law and vision system are simulated to corroborate the closed-loop system's stability and for test our result in real world some platforms have been developed, proposing a new quasi-virtual system that merge virtual world with real platform.

Résumé

Cette thèse présente la conception et la mise en œuvre pratique d'un système de navigation intérieure d'un Drone en utilisant un système de vision dont les données sont obtenues à partir des informations sur l'environnement acquises par une caméra embarquée sur un Drone. Les techniques récemment utilisent les systèmes mono vision, vision stéréo et SLAM. Dans cette thèse nous développons de techniques pour la mono vision, en raison de la charge utile limitée dans un Drone. La Fig. 1 montre le but principal de la thèse. Nous pouvons regarder l'image d'un couloir ou nous désirons faire la navigation d'un système quadrotor en utilisant que l'information obtenue d'une camera. Nous avons développé un système de vision capable d'obtenir la géométrie de l'environnement. Normalement dans un couloir nous pouvons faire l'extraction de plusieurs lignes. Ces lignes sont utilisées pour faire d'abord l'estimation de la rotation du système et ensuite grâce à la perspective de l'image obtenue pour la camera nous avons utilisé la colinéarité des lignes, qui sont trouvées dans l'intersection du sol et le plafond, pour estimer certaine position du système par rapport au couloir.

Système de vision

Les propriétés de projection en perspective dans le traitement d'images sont utilisées pour concevoir le système de vision embarquée qui vise à extraire des informations visuelles. Afin de réaliser la navigation au milieu d'un couloir, un système de vision en utilisant l'image obtenue par camera d'un couloir est utilisé dont directions orthogonales extraites par des points de fuite nous donnent certaine information de la rotation et la position du système par rapport au environnement. Dans la Fig. 2 l'algoritme general de notre systeme de vision est représenté.

La stratégie montre une séquence d'étapes pour estimer une position en fonction de l'environnement. Tout d'abord, l'image est acquise à l'aide d'une seule caméra. Ensuite, un détecteur de lignes est appliqué pour obtenir les lignes décrivant l'environnement. Ensuite, les lignes détectées sont regroupées en fonction de la pente, ce qui donne trois

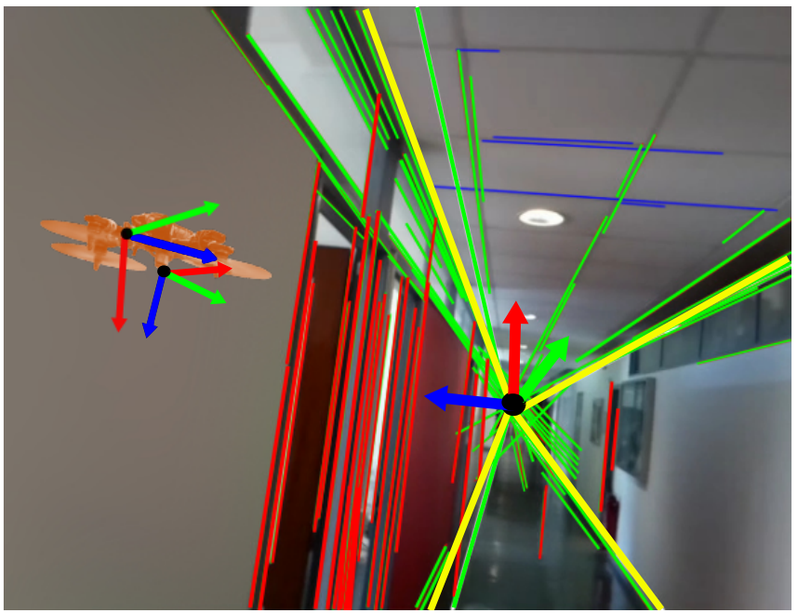


Figure 1 – Quadrotor est en train voler dans un couloir; la direction de la tête est donnée par le point de fuite, et la pose du quadrotor est obtenue géométriquement à l'aide de lignes de bord qui sont estimées à l'aide de données provenant des lignes de bords.

sous-ensembles de lignes (verticale, horizontale et diagonale). Ensuite, à partir des lignes classées, il est possible d'extraire deux points de fuite infinis et un point de fuite fini. Puis, la matrice de rotation de la caméra est estimée en utilisant les points de fuite. Ensuite, un ajustement général des moindres carrés est appliqué pour extraire quatre lignes des bords. Enfin, la colinéarité dans les lignes des bords opposées est appliquée pour estimer la pose de la caméra sur l'image plane. Cette colinéarité donne certaines valeurs en fonction de la position de la caméra dans l'espace 3D d'un couloir.

Classification des lignes et point de fuite

La Fig. 3 montre la classification des lignes. D'abord, nous pouvons observe l'entrée de l'image obtenue d'un caméra. Ensuite il est appliqué un algorithme d'extraction des lignes. Finalement, nous avons utilisée l'équation (1) pour faire la classification des lignes verticales, horizontales et verticales.

$$\mathbf{L} \quad = \quad \begin{cases} 110 \leq \mathbf{L} \geq 80 & L_v = \mathbf{L} \\ -10 \geq \mathbf{L} \leq 10 & L_h = \mathbf{L} \\ \text{autrement} & L_d = \mathbf{L} \end{cases} \tag{1}$$

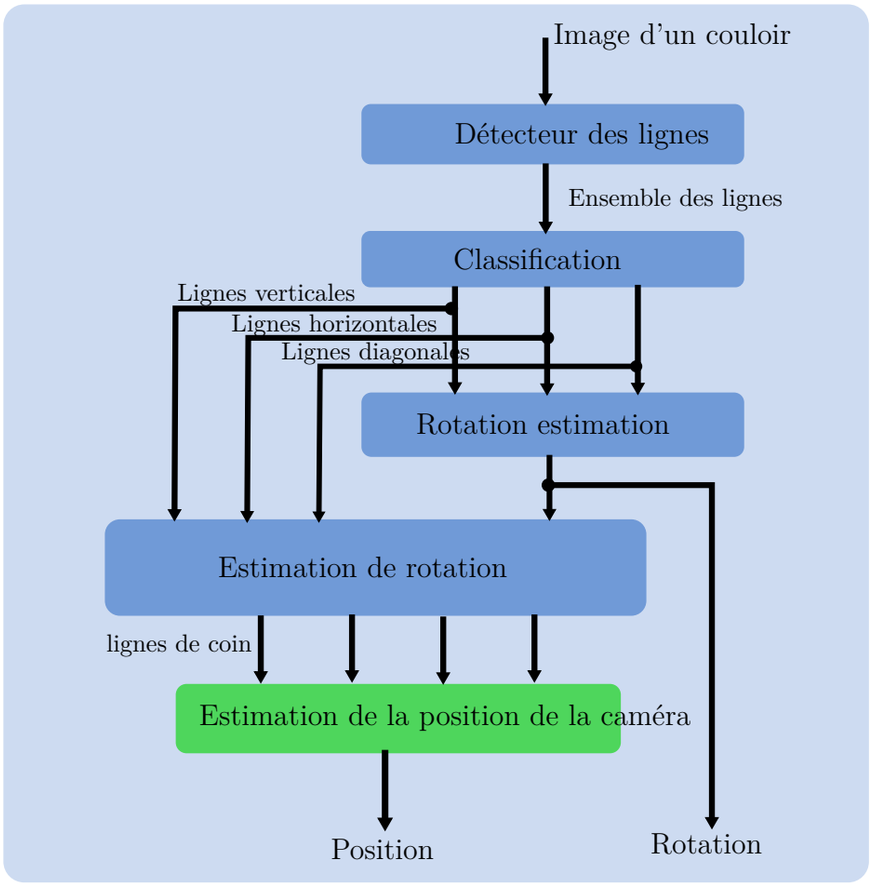


Figure 2 – Pipeline de l'estimation de la position de la caméra.

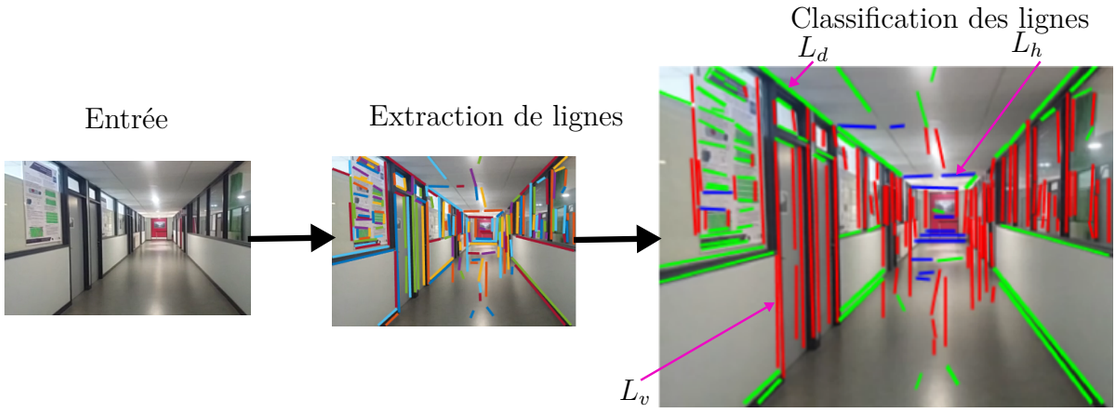


Figure 3 – Classification des lignes en fonction de la pente de chaque ligne. Les lignes verticales sont représentées en rouge, les lignes horizontales en bleu et les diagonales en vert.

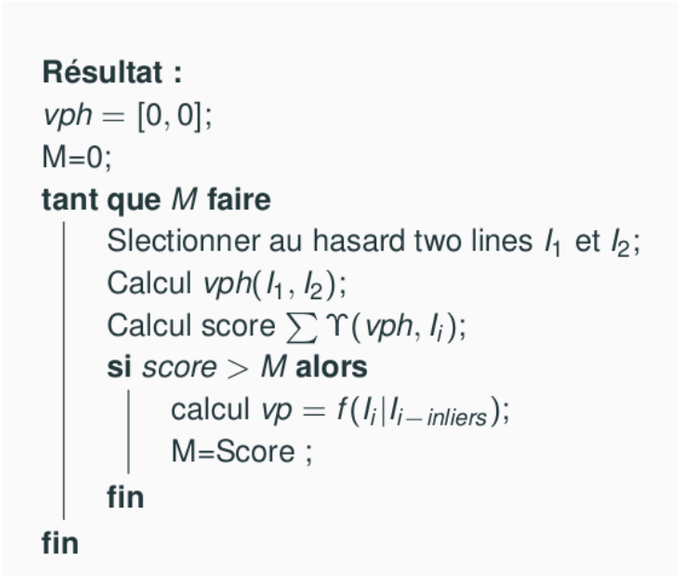


Figure 4 – Algorithme RANSAC pour eliminer les lignes aberrantes.

Les ensembles des lignes sont utilisées pour faire l'estimation des points de fuite.

Point de fuite fini

La méthode RANSAC est utilisée pour éliminer les lignes aberrantes et garder les lignes qui nous permettront d'obtenir un point de fuite (vp) fini ou infini. Dans la Fig. 4 est représentée la méthode RANSAC pour estimer les points de fuite. D'abord, Ils sont sélectionnés au hasard deux lignes dans l'ensemble des lignes soit verticale, soit horizontale ou diagonale. Ensuite il est calculé un score dans chaque ensemble en utilisant (2), (3) pour un point de fuite fini. Après, Nous avons fait une comparaison de le score pour vérifier que le pourcentage de l'ensemble des lignes soit plus grande d'une certaine valeur M. Finalement avec les ensemble des lignes basée dans le score il est calculé le point de fuite fini.

$$score = \sum_{i=0}^{n} \Upsilon(vp_h, l_i) \tag{2}$$

où n est le nombre de lignes dominantes du sous-ensemble $(L_d, L_h, L_v,)$ et

$$\Upsilon(vp_h, l_i) = \begin{cases} 1 & d(vp_h, l_i) < \delta \\ 0 & \text{autrement} \end{cases} \tag{3}$$

$d(vp_h, l_i)$ est la distance euclidienne du candidat fini vp candidat vp_h à la ligne l_i $(L_d, L_h, L_v,)$. δ est un seuil fixe donné en pixels.

Point de fuite infini

Pour trouver le point de fuite infini il est utilisée le même algorithme 4 mais l'équation (4) est utilisée pour faire le calcul de le score.

$$\Upsilon(v, l_i) = \begin{cases} 1 & Min(\widehat{(\overrightarrow{v}, \overrightarrow{l_i})}, \widehat{(\overrightarrow{l_i}, \overrightarrow{v})}) < \iota \\ 0 & \text{autrement} \end{cases} \quad (4)$$

où $\widehat{(\overrightarrow{v}, \overrightarrow{l_i})}$ est l'angle parmi vp l'infinie direction depuis le centre de l'image, et la ligne l_i est utilisée pour tester l'espace image. Dans ce cas, le consensus utilise une distance angulaire entre la direction de la droite candidate et la direction représentant le point de fuite infini. Le seuil ι est donné en degrés.

Matrix de la rotation

L'estimation de la rotation a été estimée en utilisant un point de fuite fini et deux points de fuite infinis. La Fig. 5 montre une représentation succincte de l'image d'un couloir dont lignes représentent les bords des murs, des fenêtres, des ports. Dans l'image nous pouvons regarder:

- Point de fuite vp fini
- $\vec{I_1}$ et $\vec{I_2}$ sont les directions de vp infini
- C centre de la caméra ou origin
- x_c,y_c,z_c axes du référentiel de coordonnées de la caméra
- f (distance focale)
- N référentiel de coordonnées du monde
- $[\vec{e}_1, \vec{e}_2, \vec{e}_3]$ est la rotation du monde réel par rapport à la caméra

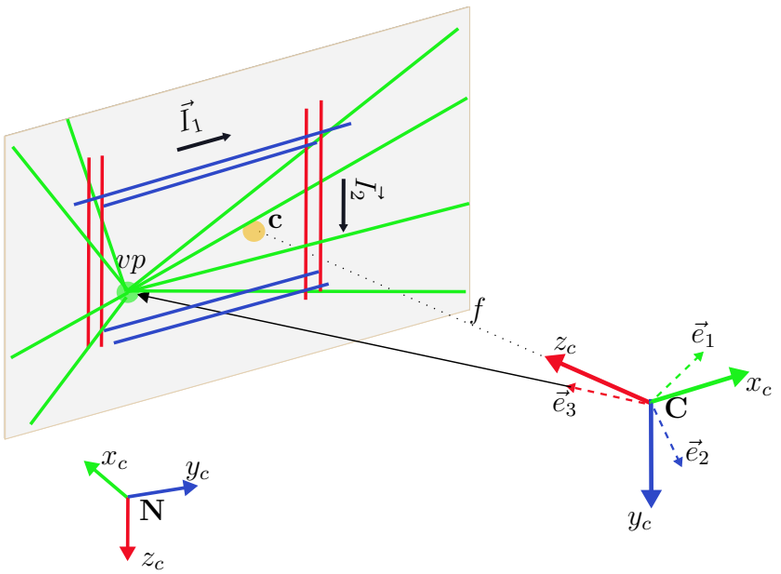


Figure 5 – Estimation de point de fuite fini.

En utilisant l'équation (5) nous pouvons obtenir le troisième vecteur de la matrice de rotation.

$$\begin{aligned} \overrightarrow{Cvp} = e_3 &= [e_{3_x}, e_{3_y}, e_{3_z}]^T \\ &= [vp_x, vp_y, -f]^T \\ \vec{e}_3 &= \frac{e_3}{||e_3||} \end{aligned} \tag{5}$$

Pour faire l'estimation de premier et deuxième vecteur de la matrice de rotation, nous utilisons les Fig. 6 et Fig. 7 pour représenter les paramètres du système. Les équations (6), (7), (8) et (9) sont utilisées faire le calcul des vecteurs manquant.

$$\begin{aligned} \vec{I}_1 &= [I_{1x}, I_{1y}, C]^T \\ e_1 &= [I_{1_x}, I_{1_y}, e_{1_z}]^T \\ e_1 \cdot e_3 &= 0 \end{aligned} \tag{6}$$

$$\begin{aligned} e_1 &= [I_{1_x}, I_{1_y}, \tfrac{I_{1_x} e_{3_x} + I_{1_y} e_{3_y}}{f}]^T \\ \vec{e}_1 &= \tfrac{e_1}{||e_1||} \end{aligned} \tag{7}$$

$$\begin{aligned} \vec{I}_2 &= [I_{2x}, I_{2y}, C]^T \\ e_2 &= [I_{2_x}, I_{2_y}, e_{2_z}]^T \\ e_2 \cdot e_3 &= 0 \end{aligned} \tag{8}$$

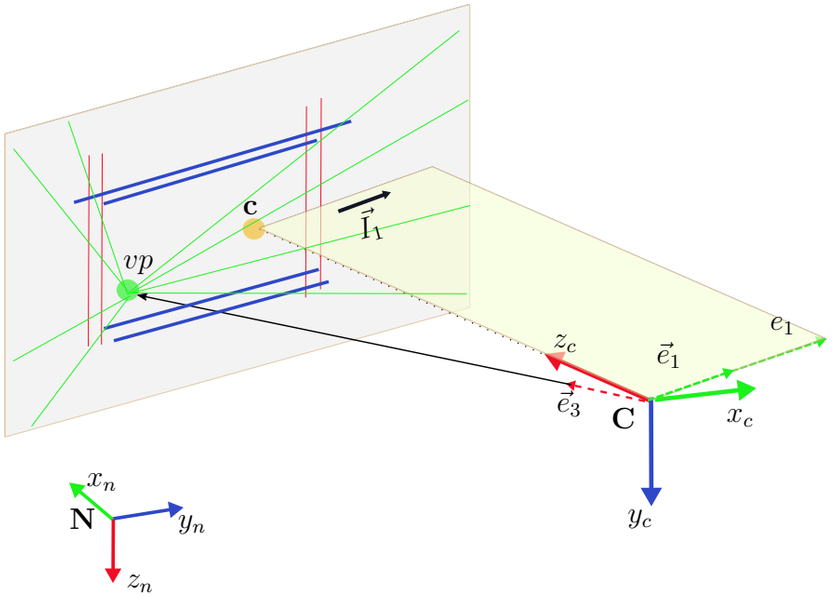


Figure 6 – Estimation de point de fuite infini, lignes horizontales.

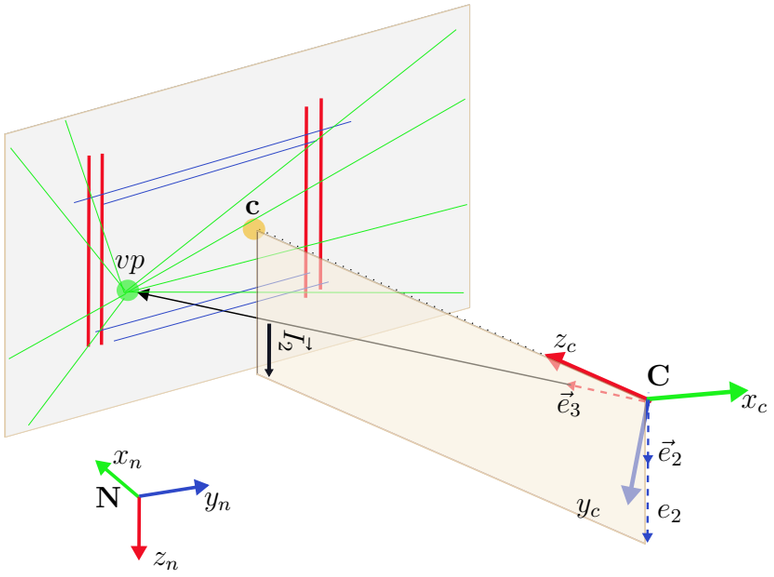


Figure 7 – Estimation de point de fuite infini, lignes verticales.

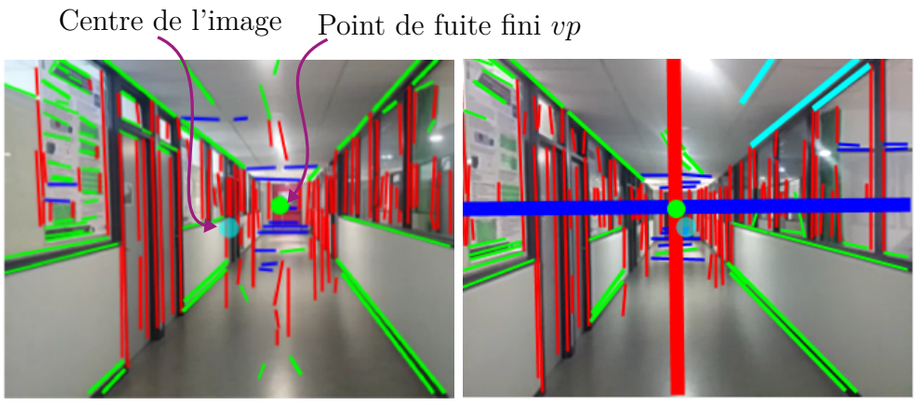


Figure 8 – Information obtenue avec l'estimation de la rotation de la camera.

$$e_2 = [I_{2_x}, I_{2_y}, \frac{I_{2x}e_{3x}+I_{2y}e_{3y}}{f}]^T$$
$$\vec{e_2} = \frac{e_2}{||e_2||} \tag{9}$$

Extraction des lignes de bord

Avec l'information obtenue avant, nous avons les donnes (Fig. 8):

- 4 quadrants
- Chaque quadrant a 3 sous-ensembles de lignes:
  verticales, horizontales et diagonales

La méthode des Moindres Carrés Généralisés est utilisée pour trouver les lignes de bord dans l'environnement structuré.

Moindres Carrés Généralisés

Une fois que nous avons obtenus les points (x_i, y_i) qui représentent les coordonnées initiales et finales des lignes de chaque quadrant, la matrice de covariance est donnée par

$$C_m \equiv \begin{bmatrix} \omega_i \sum x_i^2 & \omega_i \sum x_i y_i \\ \omega_i \sum x_i y_i & \omega_i \sum y_i^2 \end{bmatrix}$$

C_m représente la matrice de covariance, pour chaque image m, ω_i est une ponderation qui est utilisé pour eliminer les points qui dépassent une certaine limite de la ligne précédente

$$\omega_i = e^{\frac{-0.5\mu_i^2}{\sigma^2}}$$

$$\mu_i = \arctan \frac{x_i a_{m-1} + y_i b_{m-1}}{x_i a_{m-1} - y_i b_{m-1}}$$

a_{m-1}, b_{m-1} représente les parametres de ligne initiale précédente.

L'equation (10) represent l'expression pour calculer la ligne de chaque quadrant.

$$y = LQ_k = -\frac{b_m}{a_m} x \tag{10}$$

où LQ_k représente la ligne principale de chaque quadrant avec $k = 1, 2, 3, 4$

Lorsque la caméra est au centre du couloir, une colinéarité entre les lignes L_{Q_1}, L_{Q_3} et L_{Q_2}, L_{Q_4} est représentée par

$$CP_{1,3} = L_{Q_1} \wedge L_{Q_3} \tag{11}$$

$$CP_{2,4} = L_{Q_2} \wedge L_{Q_4} \tag{12}$$

et la position relative par rapport au produit vectoriel

$$y_r = sign(CP_{1,3}(3))||CP_{1,3}|| - sign(CP_{2,4}(3))||CP_{2,4}|| \tag{13}$$

$$z_r = sign(CP_{1,3}(3))||CP_{1,3}|| + sign(CP_{2,4}(3))||CP_{2,4}|| \tag{14}$$

Où y_r et z_r sont les positions relatives de la caméra par rapport au centre du couloir

Modèle du système

Pour commander et stabiliser notre système de type Drone, un schéma de commande à base de quaternion borné est présenté, celui stabilise l'orientation du Drone, et il est également utilisé pour commander sa direction, en fusionnant les informations visuelles et celles provenant de la centrale inertielle. La Fig. 9 représente le système quadrotor et les équations (15) et (16) represent le modèle du drone a six degrés de liberté.

$$\Sigma_T \quad : \quad \begin{cases} \dot{\vec{\xi}} = \vec{r} \\ m\dot{\vec{r}} = mg\vec{e}_3^n - F_u R^T \vec{e}_3^n \end{cases} \tag{15}$$

$$\Sigma_R \quad : \quad \begin{cases} \dot{q} = \dfrac{1}{2}\Xi(q)\ \vec{\eta} \\ J\dot{\vec{\eta}} = -[\vec{\eta}]_\times J\vec{\eta} + \vec{\Gamma} + \vec{\Gamma}_G \end{cases} \tag{16}$$

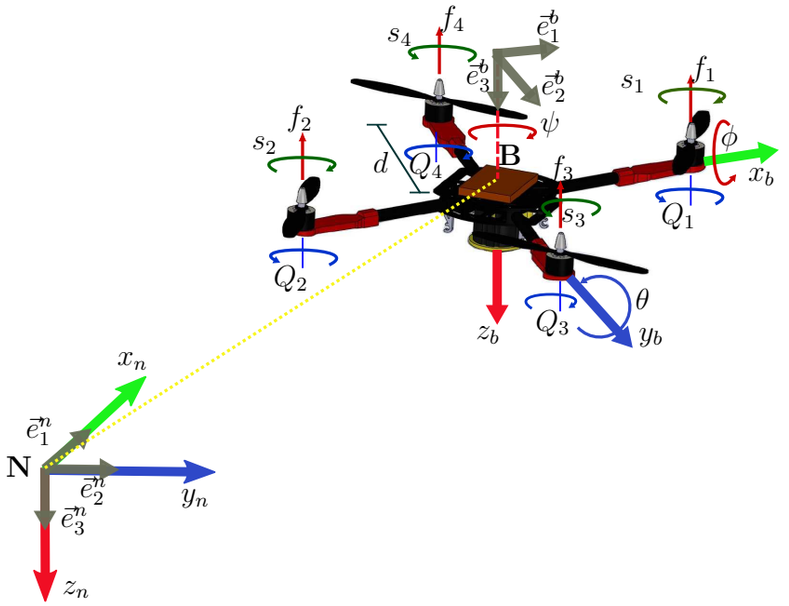


Figure 9 – Représentation d'un système quadrotor

Commande bornée en position

Nous allons commencer avec une définition pour spécifier le type de la loi de contrôle obtenue. Considérons la dynamique de rotation du corps rigide décrite par (16) avec les entrées de contrôle limitées suivantes $\Gamma = [\Gamma_\phi, \Gamma_\theta, \Gamma_\psi]]^T$, dont l'indice est modifié pour exprimer les équations suivantes, définies par:

$$\Gamma_i = -\sigma_{\overline{\Gamma}_i}\left(\frac{k\vec{\eta}_i}{\rho_i} + sign(q_0)k\vec{q}_i\right) \tag{17}$$

$\overline{\Gamma}_i$ avec $i \in 1, 2, 3$ représente la limite physique sur le torque i - th Γ_i. q_0 représente la partie scalaire du quaternion et $\vec{q}_i$ le vecteur quaternion. $\vec{\eta}_i$ spécifie la vitesse angulaire. k marque un paramètre réel tel que $0 < k \leq min_i\overline{\Gamma}_i/2$. ρ_i est un paramètre réel strictement positif.

Commande bornée en position

Supposons maintenant qu'en utilisant la loi de contrôle (17), on puisse stabiliser la dynamique de lacet, c'est-à-dire $\psi = 0$, donnée par la direction de la tête obtenue par le système de vision selon notre configuration. Ensuite, après un temps suffisamment long, le système (15) devient,

$$\begin{bmatrix} \dot{x}_n \\ \dot{y}_n \\ \dot{z}_n \end{bmatrix} = \begin{bmatrix} r_x \\ r_y \\ r_z \end{bmatrix}, \tag{18}$$

$$\begin{bmatrix} \dot{r}_x \\ \dot{r}_y \\ \dot{r}_z \end{bmatrix} = \begin{bmatrix} -\frac{F_u}{m}\sin\theta \\ \frac{F_u}{m}\sin\phi\,\cos\theta \\ \frac{F_u}{m}\cos\phi\,\cos\theta - g \end{bmatrix}, \tag{19}$$

Avec un choix approprié de ces configurations de cibles, il sera possible de transformer (18) et (19) en trois intégrateurs triples linéaires indépendants. Pour cela, prenez-nous

$$\begin{aligned} \phi_d &:= \arctan\left(\frac{\varsigma_2}{\varsigma_3 + g}\right), \\ \theta_d &:= \arcsin\left(\frac{-\varsigma_1}{\sqrt{\varsigma_1^2 + \varsigma_2^2 + (\varsigma_3 + g)^2}}\right) \end{aligned} \tag{20}$$

$$F_u = m\sqrt{\varsigma_1^2 + \varsigma_2^2 + (\varsigma_3 + g)^2} \tag{21}$$

où $\varrho = [\int x_n, x_n, r_x, \int y_n, y_n, r_y, \int z_n, z_n, r_z]^T = [\varrho_1, \varrho_2, \varrho_3, \varrho_4, \varrho_5, \varrho_6, \varrho_7, \varrho_8, \varrho_9]^T$, puis (18)-(19) devient:

$$\Sigma_x : \begin{cases} \dot{\varrho}_1 = \varrho_2 \\ \dot{\varrho}_2 = \varrho_3 \\ \dot{\varrho}_3 = \varsigma_1 \end{cases} \tag{22}$$

$$\Sigma_y : \begin{cases} \dot{\varrho}_4 = \varrho_5 \\ \dot{\varrho}_5 = \varrho_6 \\ \dot{\varrho}_6 = \varsigma_2 \end{cases} \tag{23}$$

$$\Sigma_z : \begin{cases} \dot{\varrho}_7 = \varrho_8 \\ \dot{\varrho}_8 = \varrho_9 \\ \dot{\varrho}_9 = \varsigma_3 \end{cases} \tag{24}$$

La Commande bornée en position est:

$$\begin{aligned}
\varsigma_1 &:= -\vartheta_1\{a_3\sigma_{M_1}[\frac{1}{\vartheta_1}(a_2\varrho_1+\varrho_2+\varrho_3)]+a_2\sigma_{M_1}[\frac{1}{\vartheta_1}(a_1\varrho_2+\varrho_3)]+a_1\sigma_{M_1}[\frac{1}{\vartheta_1}(\varrho_3)]\}\\
\varsigma_2 &:= -\vartheta_2\{b_3\sigma_{M_1}[\frac{1}{\vartheta_2}(b_2\varrho_4+\varrho_5+\varrho_6)]+b_2\sigma_{M_1}[\frac{1}{\vartheta_2}(b_1\varrho_5+\varrho_6)]+b_1\sigma_{M_1}[\frac{1}{\vartheta_2}(\varrho_6)]\}\\
\varsigma_3 &:= -\vartheta_3\{c_3\sigma_{M_1}[\frac{1}{\vartheta_3}(c_2\varrho_7+\varrho_8+\varrho_9)]+c_2\sigma_{M_1}[\frac{1}{\vartheta_3}(c_1\varrho_8+\varrho_9)]+c_1\sigma_{M_1}[\frac{1}{\vartheta_3}(\varrho_9)]\}
\end{aligned} \tag{25}$$

$$\begin{aligned}
\vartheta_1 &= \bar{\varsigma_1}/(a_1+a_2+a_3),\\
\vartheta_2 &= \bar{\varsigma_2}/(b_1+b_2+b_3),\\
\vartheta_3 &= \bar{\varsigma_3}/(c_1+c_2+c_3)
\end{aligned} \tag{26}$$

Ensuite, les lois de contrôle dans (25) stabiliser de manière exponentielle les systèmes (18)-(19) à la position désirée $(\varrho_1, \varrho_2) = (\varrho_{dx}, 0)$, $(\varrho_3, \varrho_4) = (\varrho_{dy}, 0)$ et $(\varrho_5, \varrho_6) = (\varrho_{dz}, 0)$.

L'objectif de stabilisation est l'origine. Dans le cas où la condition asymptotique est différente de l'origine, les variables $\varrho_2, \varrho_5, \varrho_8$ doivent être remplacées dans la loi de contrôle (25) par $e_1 = \varrho_2 - \varrho_x^d$, $e_2 = \varrho_5 - \varrho_y^d$, $e_3 = \varrho_8 - \varrho_z^d$, respectivement. Dans ce cas, $\varrho_x^d, \varrho_y^d, \varrho_z^d$ représentent la position souhaitée dans l'espace.

Évaluation expérimentale

L'estimation de la position d'un quadrotor dans les axes y et z par rapport à la référence du monde sont utilisées comme entrée pour le commande de la position bornée afin de la poser à la position souhaitée.

Des tests expérimentaux ont été effectués pour valider l'algorithme de vision en utilisant la colinéarité entre les lignes. Pour obtenir le meilleur résultat avec l'algorithme de vision, il doit être exécuté lorsque le véhicule est en train voler. La coordonnée x et le mouvement de la hauteur sont fixés pour les limitations de l'espace de travail. Alors le quadrotor se déplacera comme un avion PVTOL évoluant dans le plan $z-y$ avec un angle de roulis, et l'angle ψ prendra la direction du point de fuite qui a une valeur proche de zéro. Cela signifie que les états x et θ sont stabilisés avec le contrôleur utilisant les mesures provenant du système Vicon ou contrôlées par un pilote externe. Le véhicule se trouve au milieu du couloir virtuel lorsque la position fournie par le système Vicon est à $x = 0$, $y = 0$ et $z = 0.9$, tous en mètres. Fig. 10 montre le schème pour le système quasi-virtuel que nous avons développé.

Dans la Table 1 est représentée les positions et les temps que nous avons utilisés dans

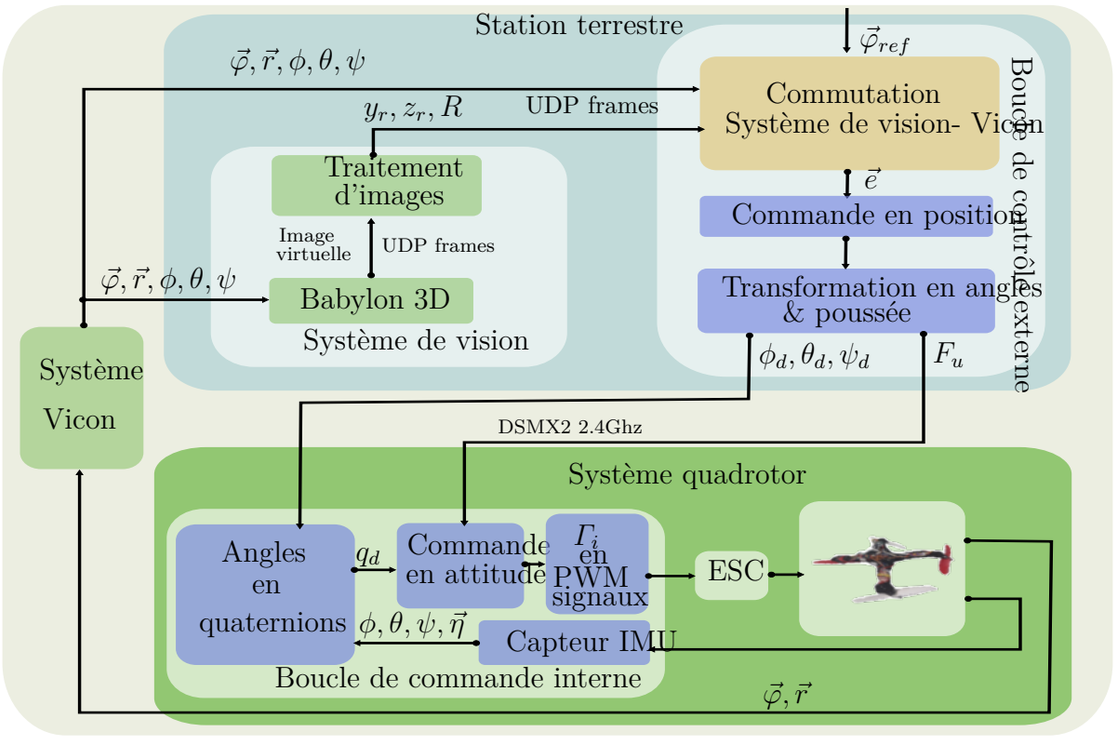


Figure 10 – Système quasi virtuel

ce test

Table 1 – Position aux tests quasi virtuels

Position initiale	1	2	3	4	5	6	7	8
Temps (s)	0	12	75	85	110	185	163	170
y	0.5	0	-0.6	0	0.6	0	-0.6	0.0
z	0.5	0.9	1.3	0.9	1.3	0.9	0.6	0.9

Dans le Fig. 11 le comportement du système et la Fig. 12 est représente le système de vision et véhicule fonctionnant avec une image virtuelle.

Il convient de mentionner que la stratégie de vision n'est pas capable en mesure d'estimer l'axe des x, donc cet axe est contrôlé d'une façon manuel.

Conclusions

- Le schéma de commande utilise la fusion données provenant de plusieurs capteurs et en temps réel: IMU, caméra et Salle Vicon.

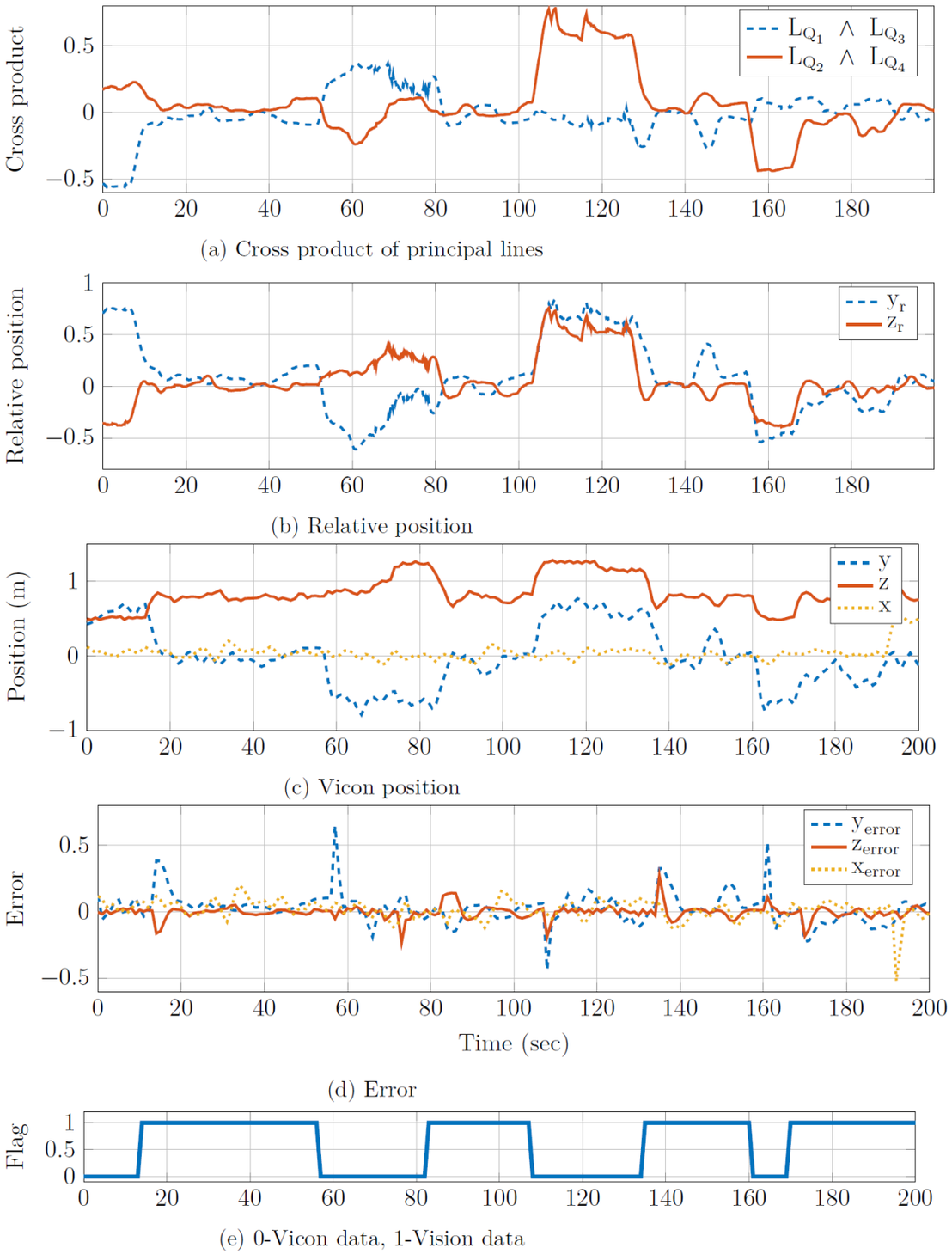


Figure 11 – Stabilisation à l'aide du système Vicon, véhicule et de Babylon 3D.

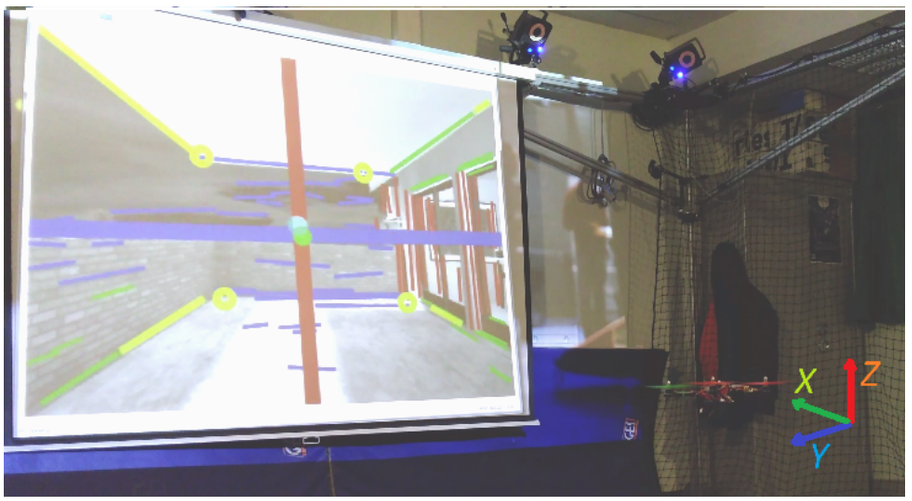


Figure 12 – Système de vision et véhicule fonctionnant avec une image virtuelle.

- Le système de navigation a été testé et validé sur une plateforme expérimentale de type quadrotor
- Vision (traitement d'images, détection de bords, points de fuite)
  - Obtention de la matrice de rotation grâce aux points de fuite
- Commande bornée sur l'attitude (PD) et la position (PID)
- Commande pour l'évitement de collisions avec l'aide des lignes de bord
- Validation expérimentale du système sur plusieurs configurations: Simulation, Quasivirtuel et Réel.

Travaux futurs

- Utiliser un filtre de Kalman pour filtrer les données obtenu et pour estimer les vitesses linéaires
- Concevoir une stratégie pour compenser les retards des données estimées (prédicteur de Smith)
- Développer le système complet dans une plateforme embarquée

Introduction

The localisation problem in indoor environments has been subject to interest by both industry and academy last years. If global positioning system (GPS) sensor can be used in outdoor environment, to navigate in indoor environments one has to take different kind of technologies due to the unavailability or at least weak reception/resolution of GPS measurements. Thus, several strategies have been proposed in the literature to solve this problem, among which vision systems have become the most adopted strategy to complete this task. The vision system is able to extract the characteristics of an environment, which can be used to estimate the position of the system. It means that shape, colour, geometry of the environment can be used to extract the rotation and position of a camera, which is usually on-board of a robotic system (in this work a quadrotor).

In urban scenes, buildings are aligned on rectangular grids which means only three mutually orthogonal directions exist in a scene, named Manhattan world in Coughlan and Yuille [1999]. These orthogonal directions can be extracted using the lines that describe the shape of the environment. Windows, doors and some furniture can be also described by lines. Particularly in a corridor, as illustrated in Fig. 13, walls, windows, and doors are described by vertical and horizontal lines. Roof and floor are represented with horizontal lines, and the depth of corridor described by diagonal lines. It can be observed that all parallel lines in real world converge to a single point in the image plane to camera perspective. In practical applications, we usually find more than one vanishing point due to the imperfection of lines extraction and calibration of the camera. These points are known as vanishing points *vp*s, and can be used for camera calibration Caprile and Torre [1990a]. Taking into account that the camera is placed in front of the quadrotor, the position of the vanishing points in the image can be used to estimate the head direction of the robotic system, see Fig. 13. In addition, all the extracted lines which can be acquired by using the image of a corridor, are normally either concentrated or intersected on the edges. Since a line can be described by a sequence of points, the edge of a corridor can be represented by a neighbourhood of points describing the intersection between the floor and roof. Thus, these points can be used to extract four lines describing the edges of the corridor.

Keeping in mind that the quadrotor's head direction has to be maintained pointing to the vp, collinearities between edges lines can be used to extract pose informations, that is the position of the vehicle relative to the center of the corridor. These data is estimated with respect the geometry of the environment (corridor). Nevertheless, only horizontal and vertical positions (y, z-axes with respect to the real world) can only be estimated using this approach.

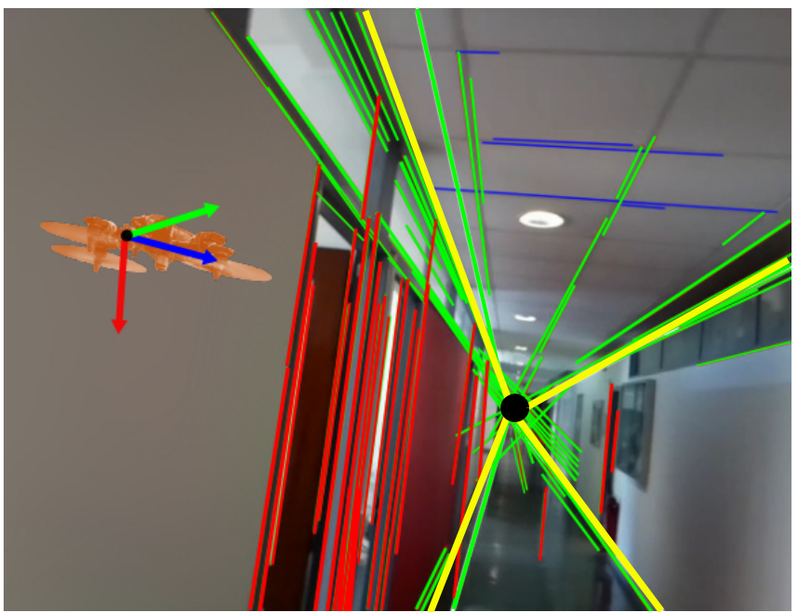


Figure 13 – Quadrotor flying through a corridor; the heading direction is given by a vanishing point, and the pose of quadrotor is obtained geometrically using edges lines which are estimated using data from the corner borders.

This thesis is focused on designing a strategy to partially solve the localization problem of a quadrotor vehicle using images from a single on-board camera. The proposed algorithm extracts information from images taken by a camera in order to pose the quadrotor in the middle of a corridor. More specifically, the different stages of our proposed solutions are:

1. To design an algorithm to estimate the position in a corridor using the video image from an embedded camera.
2. To propose an attitude control law for quadrotor vehicle.
3. To develop a position control law for quadrotor vehicle.
4. To plan a quasi-virtual system that includes a real system and an image video of a 3d engine are combined to validate the tests.
5. To perform simulations and real test experiments.

Overviews

This thesis is organized in 6 chapters. The three first ones present the state of art and the background of vision and quadrotor systems. The remaining of the work describes the proposed solution and the experimental tests. Below a brief summary of each chapter is given:

Chapter 1

Since this work is focused on two systems: vision and control, this chapter has been centered on analyzing the state of the art of both quadrotor vehicle and vision systems. First, some works on the mathematical modelling of a quadrotor system are given. Then a description of several control laws regarding attitude and position are provided. Next, the problem of localization is described, and some works about indoor navigation of quadrotor vehicle using a vision system are mentioned. Finally, some works which motivated the idea of extracting data from structured environment are described.

Chapter 2

This chapter is focused on describing some concepts about vision systems. Points, lines joining two point lines, intersections of two lines, etc. are defined. The concept of estimating the calibration of a camera are detailed. Then, projective projections are explained in order to see how three vanishing points can depict a structured environment. Finally a simple algorithm to estimate the vanishing point using extracted lines from an image is presented.

Chapter 3

This chapter is devoted to present some mathematical preliminaries and concepts for a quadrotor system. Firstly, problems of rigid body, attitude, and position representations are discussed. Moreover, a presentation of the different configurations for aerial vehicles is shown by highlighting the importance of the VTOL aircrafts for some specific applications. In the second part, the relation between world, body, and camera frames to merge a complete system is described. All these concepts will be used along the present work. In the third part, a review about the works, which proposed some attitude and position control laws for UAVs, and considering that localization problem has been resolved, is reported. Then some works, taking into account that the localization is solved using vision systems are mentioned. Furthermore some advantages and disadvantages are analyzed

depending on the application context and the environment. Finally, a study of different control strategies is performed, analyzing stability, robustness and tracking performance.

Chapter 4

This chapter introduces an algorithm for computing the camera pose. The strategy is based on camera calibration using the vanishing points and geometry into a corridor. First, a description of the pipeline algorithm is detailed. Then, the dominant lines detection which is employed to extract the lines from an image is presented. In a second time, the algorithm classifies all the extracted lines. Third, vanishing points are computed giving the mobile unitary frame. The rotation transformation from fixed to mobile world can be deduced giving an orthogonal matrix. Finally, camera pose is estimated by means of the edges (corner) lines, which are the lines describing a corridor edge. At the end of chapter, a numerical validation is presented in order to show the effectiveness of the algorithm, and some results from rotation matrix are compared with data obtained by a motion capture system.

Chapter 5

Chapter 5 deals with the attitude and position control laws for a quadrotor system. First, the objective is to design a control law which drives a quadrotor for attitude stabilization under bounded torques. In this chapter is also depicted two configurations to test the proposed algorithms. The first one proposes a strategy to make a navigation in a virtual environment. The second one merges a virtual environment with a real quadrotor, which is called in this work *Quasi-virtual* system. At the end, a simulation test is presented showing the effectiveness of our algorithms.

Chapter 6

The description of system is presented in Chapter 6. A review of motion capture system, ground station, features on hardware, software, communication, and platforms are presented. Then, experimental results are detailed. These tests include the quasi-virtual scenario, where are fused a virtual environment and a real quadrotor vehicle. Next, results obtained using the images acquired by on-board camera of a quadrotor vehicle are analyzed. Finally, a test made in a real corridor is presented.

Finally conclusions and future works are discussed in **Chapter 7**.

Chapter 1

State of the art

Indoor navigation of a quadrotor vehicle is a task which can be performed using several strategies. The research community and the industry have a great interest in developing Unmanned Aerial Vehicle (UAV) in order to solve different problems that can be tackled using these systems. Even in outdoor environments, if the position's estimation problem can often be solved using GPS, there are some tasks where GPS data are unavailable, because these data are often disturbed or noised. Therefore, this problem is becoming a challenge for developing new strategies to estimate the pose system without GPS data.

One solution is to fuse measures from magnetometers, accelerometer and digital compass using observers algorithms like Kalman filters. However, the problem with this method is the fusion process resulting in large cumulative drift errors due to the coarsity of the sensors. Some approaches have tried to solve the estimation problem using, Time of Flight system (TOF) (Gokturk et al. [2004],Ximenes et al. [2018]), Kinect (Eric and Jang [2017], Vadakkepat et al. [2016]), and on-board cameras. Simplicity, efficient distance algorithm, and speed are some advantages of TOF systems; however due to the physics of the sensors, light and interferences are their principal disadvantages. The Kinect sensor is an infrared projector, a camera, and a special microchip that generates a grid from which the location of a nearby object in 3D can be ascertained. This information can be used in an indoor navigation system, but the size and weight of these sensors make it difficult to install in lightweight systems. Thus, on-board cameras are a good option to deal with these problems. A camera is lightweight, and the image acquired yields a lot of useful information to estimate the position system with respect to the real world.

This chapter presents a review of the literature about quadrotors and vision systems. Some works about indoor navigation using as platform a quadrotor will be introduced. And then literature, which has been served as a tool to develop this thesis work, is presented.

1.1 Indoor navigation of a quadrotor using a vision system

Nowadays, several works using mono-vision and stereo vision system have been carried out; however, the problem is still unsolved, and many interesting works arise steadily due to the advances in embedded computer vision which offer a good choice for helping to solve the indoor localization problem. For instance, some pose estimation algorithms use only the information given by the cameras, nevertheless, others approaches combining different sensors (IMU, radar, sonar etc.) have became also popular. For example, in Schauwecker and Zell [2013] a stereo vision systems with a SLAM (Simultaneous Localization and Mapping) algorithm is used. Here the system is equipped with four cameras arranged in two stereo configurations. It is able to recover pose estimation errors and can cope with processing failures for a camera pair. In García Carrillo et al. [2012] the development of a quadrotor, able of autonomously flight indoors using mainly the combination of stereo vision and an inertial navigation system is presented. The authors applied a Kalman filter to fuse several sensors like IMU, ultrasonic, etc, which are used to estimate the angular rate and attitude. Nevertheless, the integration of these signals leads to unbounded low-frequency drift. In Bristeau et al. [2011], a localization strategy has been developed using a mono-vision system and the combination of low-cost inertial sensors. The authors used the popular AR Drone for testing. Nevertheless, a complex integration algorithm of its sensors is needed to achieve a complete navigation. In Weiss et al. [2011] the authors present a vision-based MAV control approach, where the pose is estimated by means of a monocular SLAM algorithm with a precision of few centimeters. This approach needs short period of stationary hovering to adjust the local map with the gravity vector, and a continuous flight without pauses is still an open issue. In Wang et al. [2013] a complete navigation scheme for an indoor flight using a quadrotor is proposed. The authors select, customize, and combine suitable existing algorithms (SLAM, FastSLAM) for their navigation scheme, nevertheless it needs three main sensors which are used on-board the platform, namely an inertial measurement unit, a downward-looking camera and a scanning laser range finder.

Besides these SLAM based approaches, many vision based control approaches are proposed in the literature. For instance, in Zhang et al. [2009] a vision-based method to assist landing of aircrafts is proposed, which only uses two edge lines and the threshold line on the runway. The authors describe that using this algorithm, it is possible to estimate yaw and pitch angles, and also the cross position and the altitude of the aircraft, nevertheless, simulation results have only proved its accuracy and speed. Authors in Mondragón et al. [2010] proposed a robust real-time 3D pose estimation system for UAVs using homographies. The algorithm is validated on real flights where the estimated data are compared

with the one obtained by the IMU. It has been proved that the method robustly estimates the 3D position with respect to a reference landmark, with a high quality on the position and orientation estimation when the aircraft is flying at low altitudes. A recent work is presented in McGuire et al. [2017]. In this paper a highly efficient computer vision algorithm called Edge-FS for estimating velocity and depth is presented. The algorithm uses edge distributions to calculate optical flow and stereo disparity. The velocity and depth measurements were used for fully autonomous flight of a pocket drone only relying on on-board sensors. This algorithm allows the MAV to control its velocity and avoid obstacles.

Furthermore, there are works focusing on performing an indoor navigation into a corridor. For example, a real-time monocular vision based range measurement method for Simultaneous Localization and Mapping (SLAM) applied in an UAV is presented in Celik et al. [2008]. Even though, the indoor architecture is represented via corner based feature points, the system is limited by the capabilities of the camera and the availability of good corners. In addition, the algorithm depends on the computational resources which affects the real-time quality. In Bills et al. [2011], an algorithm for autonomous navigation that uses the perspective cues was designed, but it is only able to give the desired direction for the UAV. An approach for wall collision avoidance using a depth map based on optical flow from on board camera images is presented in Simon et al. [2010], though IMU data is needed for compensating rotational effects of the optical flow.

Finally, some works proposed techniques using vanishing points; in Gomez-Balderas et al. [2011] an algorithm based on vanishing points is proposed to estimate the 3D position of a quadrotor vehicle. The results presented are in real time, but control and vision algorithms have been tested only using line painted in a wall. A similar work is presented in Wang [2011], where the ground plane vanishing line and a vanishing point are calculated from a set of equally-spaced parallel lines and rotation matrix is used in a constrained estimation of the camera location.

1.2 Extracting information from a corridor

An image, taking by a camera, can be analyzed according to its perspective, color, shape, and texture. In a common indoor environment due its classic architecture there are so many lines describing its geometry meaning wall, doors, windows, furniture, floor, corners and corridors that are formed mostly by lines (vertical, horizontal and diagonal). An image of a corridor is characterized by lines and these lines can be extracted using a vision system, see Fig. 1.1. These data can give the information to estimate the rotation and position of a camera.

Figure 1.1 – Images showing a common corridor, it can be seen that lines can describe structure, doors, and windows. Figures a),b), and c) were token with a smart-phone camera. Figure d) and e) were token using a camera FPV with a 240x320 pixels. In c) some lines describing the structure are shown.

Images in the Fig. 1.1 have in common that camera heading has been pointed in direction to the end of corridor. This guidance is useful, since to navigate along a hallway can be achieved keeping this orientation. This head-direction could be estimated by tracking three orthogonal vanishing points in a video stream. That means, once line segments in an image have been determined, vanishing points can be extracted from perspectives images.

Furthermore, when the human eye (or camera view) sees a scene of a corridor, the environment seems to converge to the end of hallway due its perspective. Such a phenomenon is characteristic in camera perspective. The view perspective produces a distortion in the image plane whose border lines have variation in its angles due to the same movements.

These variation on the slope of edges lines is used to solve partially the camera localization. These lines are formed by the union of the walls between ceiling and floor. This lines are called edges lines in this work). The collinearity of these principal lines yields a partial solution to estimate the position of a camera with respect to the center of the corridor.

This thesis work was inspired by works using a single camera and perspective projection. Below some works are described which have inspired the design of proposed algorithm. Several works have been focused on rotation matrix extraction using a strategy which vanishing points describe the three axes of the coordinate frames of the real world. For example in Zhang [2000], Caprile and Torre [1990b] some methods are proposed, which use simple properties of vanishing points. These algorithms can extract the intrinsic and extrinsic parameters. However they need a single image of a cube and two (or more) cameras (Caprile and Torre [1990b]) or a planar pattern (Zhang [2000]) to compute their parameters. A single camera method using two or three vanishing points is mentioned in Orghidan et al. [2012], Li et al. [2010], where an analysis using perspective projection is made to obtain camera parameters.

In Li et al. [2012, 2010], strategies to make vanishing point detection in man made environment are presented. However, even though its solution is designed to work on real time with a low computational cost, the presented results are only off-line.

A new method for the simultaneous computation of set of lines meeting at multiple vanishing points using the Expectation-Maximization (EM) algorithm is presented in Nieto and Salgado [2011]. In this work a vanishing point is treated as a 3D direction, instead of just a 2D point on an image (a similar strategy is described in Boulanger et al. [2006a]).

In Boulanger et al. [2006a] the authors present a walk-through inside a single image by rendering a box textured using the input data. The camera calibration is robust enough to work with non-architectural scenes, and is also capable to process several input images of the same 3D environment to navigate through this environment. The technique presented for camera parameters extraction is based on finding three vanishing points assuming that principal point of the camera is set on the center of the image. It is assumed that the directions of the three vanishing points are orthogonal. This work can be implemented on a smartphone, but it is not fast enough (0.97s) for an real time navigation of a quadrotor vehicle. This can be also seen in Gomez-Balderas et al. [2011], where vanishing points are considered as 2D point on the image plane. In practical applications these approaches suffer from many spurious parallel line segments. In Lee and Yoon [2015] a novel method is proposed that jointly estimates the vanishing points and camera orientation based on sequential Bayesian filtering. Regarding to line detection the most common strategy is using the Hough transform (Duda and Hart [1972a]), which is a powerfull tool to

identify lines. Several libraries in Matlab, OpenCV are available to accomplish this task. However, it is necessary to pre-compute the edges. In addition tuning the parameters is quite complicated and sometimes the computation cost is not low. In the last years, new strategies were proposed, as Akinlar and Topal [2011], Pătrăucean et al. [2012], to perform the edge detection. Particularly, in this work, the libraries developed in Akinlar and Topal [2011] have been applied. Those libraries require no tuning and run at the rate of 3mx for an image of 240x320 pixels.

In this work, a solution is proposed for solving the indoor autonomous aerial navigation problem. First, the proposed methodology is based on camera calibration using a vanishing point strategy. Then, the obtained information is used to find the collinearity of edge lines. Finally, the variation of the collinearity between edges lines is used as the partial UAV position.

As additional reference, in Padhy et al. [2019] was presented an approach very similar to us. They proposed a methodology using a fast procedure to estimate the set of parallel lines whose intersection would yield the position of the vanishing point (VP) inside the corridor. They also proposed a suitable measure which is formulated based on the position of VP on the intersecting lines in reference to any of the image boundary axes which helps safe navigation of the UAV avoiding collisions with side walls. However, this methodology is not capable to obtain certain position with respect to the environment, because its algorithm is not considering the geometry of environment as we proposed. In addition they used a commercial platform to make their experiments. In our work, several platforms were developed to complete all the experiments.

1.3 Summary

This chapter was focused on the state of art on quadrotor vehicle and vision systems. Firstly some of principal industry applications using drones was presented. Then, a study about the control laws is described. Next, a brief state of the art using a vision system to solve the problem of localization for indoor navigation was detailed. Finally, some works serving to extract the information on environment were analyzed focusing on strategies to extract rotation matrix using vanishing points.

Chapter 2
Preliminaries: Vision system

This chapter presents a recall of the main existing concepts of vision systems used in this work. First, the geometrical model is analysed by focusing on concepts of points, lines, translation, rotation. Then, image representation using the classic pinhole model is detailed in order to understand how to extract intrinsic and extrinsic parameters. Next, projective projections, which is the way how an image can be represented using vanishing points, is detailed. Finally a brief description about some techniques for vanishing point detection is presented.

2.1 Background

The way a point is represented in the 3D real world is usually determined by three coordinates $[x, y, z]$. In image plane, a point will be represented in 2D and is defined by two coordinates $[x_s, y_s]$. The relation between the 3D and 2D world is known as the projective projection. This process is modeled by central projection in which a ray from a point in space is drawn from a 3D real world point through the center of projection.

Points, lines, rotation, scale, translation and 2D transformation like, 2D and 3D transformations will be defined below.

Points

A pixel coordinates in an image plane is considered as a 2D point and it can be denoted using a pair of values

$$p = \begin{bmatrix} x_s \\ y_s \end{bmatrix} \tag{2.1}$$

where p represents a point and $[x_s, y_s]^T$ its coordinates in the image plane. In the sequel, this representation is called non-homogeneous by opposition to the homogeneous

representation.

Another way to illustrate 2D points consist in using homogeneous coordinates, $\tilde{p} = [\tilde{x_s}, \tilde{y_s}, \tilde{w}]^T$, where vectors that differ only by a scale factor are considered to be equivalent. An homogeneous vector $\tilde{p}$ can be converted back into its non-homogeneous form p by dividing through by the last element $\tilde{w}$.

$$\tilde{p} = \begin{bmatrix} \tilde{x}_s \\ \tilde{y}_s \\ \tilde{w} \end{bmatrix} = \tilde{w} \begin{bmatrix} x_s \\ y_s \\ 1 \end{bmatrix} = \tilde{w}\bar{p}$$

where $\bar{p} = [x_s, y_s, 1]^T$ is the augmented vector. Homogeneous points whose last element is $\tilde{w} = 0$ are called **ideal points** or **points at infinity** and do not have an equivalent non-homogeneous representation. This type of points will be detailed later.

Lines

While edges and general curves are suitable for describing contours of natural objects, the man-made world is full of straight lines. Detecting and matching these lines can be useful in a variety of applications, including architectural modelling, **pose estimation** in urban environments, and the analysis of printed document layouts. A line can also be represented using homogeneous coordinates $\tilde{l} = [a, b, c]^T$. The line equation is given by

$$\bar{p} \cdot \tilde{l} = ax_c + by_c + c = 0 \tag{2.2}$$

The normalized line equation is given by $l = [\hat{n}_x, \hat{n}_y, d]^T = [\hat{n}, d]^T$ with $\|\hat{n}\| = 1$, where $\hat{n}$ is the normalized expression of the vector $[a, b]$ and $d = c/\|[a, b]\|$. Here, $\hat{n}$ is the normal vector perpendicular to the line and d is its distance to the origin (Fig. 2.1b and Fig. 2.2). As represented on Fig. 2.1a, $\hat{n}$ can be also expressed as a function of rotation angle θ, $\hat{n} = [\hat{n}_x, \hat{n}_y] = [\cos\theta, \sin\theta]$. This representation is known as polar coordinates.

Intersection of two lines

When using homogeneous coordinates, the intersection of two lines is computed as

$$\tilde{p} = \tilde{l}_1 \times \tilde{l}_2 \tag{2.3}$$

where $\tilde{l}_1$ and $\tilde{l}_2$ represent two lines, $\times$ is the cross product operator, and $\tilde{p}$ denotes the obtained point with the intersection of two lines.

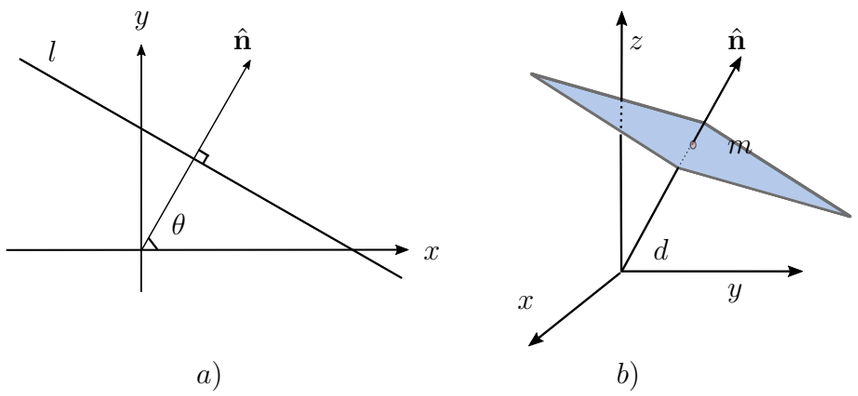


Figure 2.1 – a) 2D plane equation, expressed in polar coordinates. b) 3D plane equation, expressed in terms of the normal $\hat{n}$ and distance to the origin d.

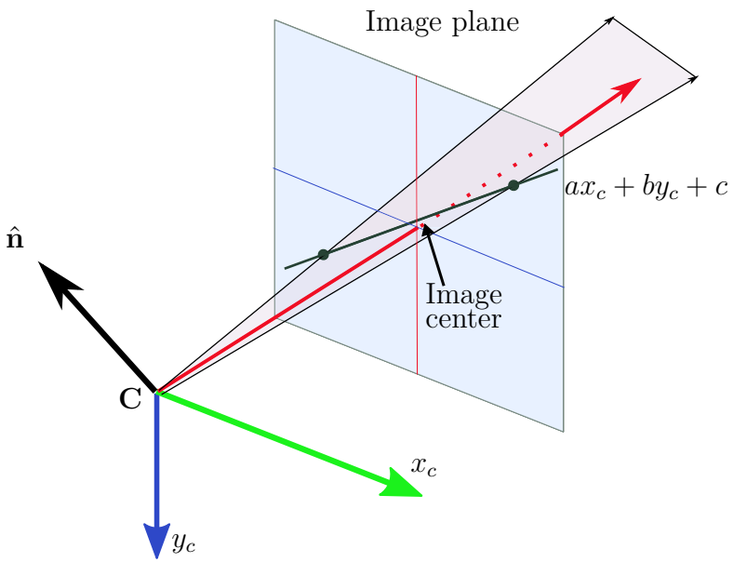


Figure 2.2 – Line projection and normal vector representation.

Line joining two points

The line joining two points of homogeneous coordinates $\tilde{p}_1$ and $\tilde{p}_2$ can be written as

$$\tilde{l} = \tilde{p}_1 \times \tilde{p}_2 \tag{2.4}$$

where $\times$ represents the cross product operator, and $\tilde{l}$ denotes the line joining the two points.

Ideal points and the line at infinity

Consider two parallel lines $ax + by + c_1 = 0$ and $ax + by + c_2 = 0$. These are represented by vectors $\tilde{l}_1 = [a, b, c_1]^T$ and $\tilde{l}_2 = [a, b, c_2]^T$ for which the first two coordinates are the same. Computing the intersection of these lines is trivial using (2.3). The intersection is $[c_2 - c_1][b, -a, 0]^T$, and ignoring the scale factor $c_2 - c_1$, this is the point $[b, -a, 0]^T$. That means that the inhomogeneous representation in this case does not exist. Except it is possible to suggest that the intersection point has infinitely large coordinates. So, points with homogeneous coordinates $[x_c, y_c, 0]^T$ do not correspond to any finite point in $\mathbb{R}^2$ but can be seen as representing a point at the infinite.

2.1.1 2D transformations

The basic transformation are described below:

Translation

2D translation of a point can be written as $p' = p + t$ or

$$p' = \begin{bmatrix} I & t \end{bmatrix} \bar{p} \tag{2.5}$$

where I is the 2×2 identity matrix, t depicts the translation vector, p is the point coordinate before the translation ($\bar{p}$ is its homogenous point), p' is the point coordinate after the translation.

Rotation + translation

It can be written as $p' = rp + t$ or

$$p' = \begin{bmatrix} r & t \end{bmatrix} \bar{p} \tag{2.6}$$

where r is a 2×2 orthonormal rotation matrix with $rr^T = I$ and $|r| = 1$.

Scaled rotation

This transformation can be expressed as $p' = srp + t$

$$p' = \begin{bmatrix} sr & t \end{bmatrix} \bar{p} \tag{2.7}$$

$s = [s_x, s_y]$ is the scale for each axis direction.

Mixing translation, rotation and a scale factor and its homogeneous representation

This expression can be expressed as $p' = srp + t$ or

$$p' = \begin{bmatrix} sr & t \end{bmatrix} \bar{p} \tag{2.8}$$

The homogeneous representation is given by

$$\bar{p'} = \begin{bmatrix} sr & t \\ 0_{1\times 2} & 1 \end{bmatrix} \bar{p} \tag{2.9}$$

where $0_{1\times 2}$ is the zero line vector of dimension 2. Using a 2×3 matrix results in a more compact notation, whereas using a full-rank 3×3 matrix makes it possible to chain transformations using matrix multiplication. Note that in any equation where an augmented vector such as $\bar{p}$ appears on both sides, it can always be replaced by a full homogeneous vector $\tilde{p}$.

2.1.2 3D transformations

3D transformations are very similar to 2D transformation, and are detailed below:

Translation

3D translations can be written as $P' = P + T$ or

$$P' = \begin{bmatrix} I & T \end{bmatrix} \bar{P} \tag{2.10}$$

where I is the 3×3 identity matrix, T depicts the 3D translation vector, P is the 3D point coordinate before the translation ($\bar{P}$ is its augmented vector), P' is the 3D point coordinate after the translation.

Rotation + translation

It can be written as $P' = RP + T$ or

$$P' = \begin{bmatrix} R & T \end{bmatrix} \bar{P} \tag{2.11}$$

where R is a 3×3 orthonormal rotation matrix with $RR^T = I$ and $|R| = 1$.

3D rotation is a non-trivial task which can be expressed in a compact way. So that, in Chapter 3 a detailed description is presented.

Scaled rotation

This transformation can be expressed as $P' = SRP + T$

$$P' = \begin{bmatrix} SR & T \end{bmatrix} \bar{P} \tag{2.12}$$

$S = [s_x, s_y, s_z]^T$ is the scale for each axis direction.

Mixing translation, rotation and a scale factor and its homogeneous representation

This expression can be expressed as $P' = SRP + T$ or

$$P' = \begin{bmatrix} SR & T \end{bmatrix} \bar{P} \tag{2.13}$$

The homogeneous representation is given by

$$\bar{P}' = \begin{bmatrix} SR & T \\ 0_{1\times 3} & 1 \end{bmatrix} \bar{P} \tag{2.14}$$

where $0_{1\times 3}$ is the zero line vector of dimension 3. Using a 3×4 matrix results in a more compact notation, whereas using a full-rank 4×4 matrix makes it possible to chain transformations using matrix multiplication.

2.1.3 Camera calibration

Concepts mentioned above are the basis for each element on the camera sensor matrix whose intrinsic and extrinsic parameters explain the relation between the real word and the image plane. The graphic illustration using the classic pinhole representation is detailed below.

Image through the pinhole

If the point P with coordinates $P = [x_c, y_c, z_c]^T$ in the referential centered with respect to the optical center C, and the z_c axis is parallel to the optical axis (of the lens), then from similar triangles, the coordinates of P and its image p are linked by the ideal perspective projection, see Fig. 2.3:

$$x_s = -f\frac{x_c}{z_c}, \quad y_s = -f\frac{y_c}{z_c} \tag{2.15}$$

where f is the focal length and $[x_s, y_s]^T$ represents the coordinates in the image plane.

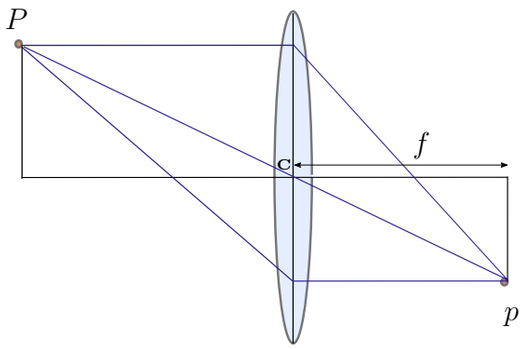


Figure 2.3 – The image of the point P is the point p where the radius parallel to the optical axis and the ray that crosses the optical center $\mathbf{C}$ intersects.

The negative sign of Eq. (2.15) means that the image appears inverted on the image plane. To eliminate this effect, the image can be flipped $[x_c, y_c] \mapsto [-x_c, -y_c]$, this corresponds to placing the image plane $z_c = -f$ in front of the optical center instead of $z_c = +f$. Thus it is considered as the frontal pinhole model in the literature (see Fig. 2.4). In that case, one has:

$$x_s = f\frac{x_c}{z_c}, \quad y_s = f\frac{y_c}{z_c} \tag{2.16}$$

In practice, the size of the image plane is bounded, therefore all points p located in the real world do not have an image in the image plane. Therefore, the field of vision is defined as the angle subtended by the spatial extent of the sensor, from the optical center.

Camera Matrix

Once a 3D point has been projected through an ideal pinhole using a projection matrix, the resulting coordinates must be transformed according to the pixel sensor spacing and the relative position of the sensor plane to the origin. Then this transformation is given by:

$$\tilde{p} = K \begin{bmatrix} R & T \end{bmatrix} \bar{P} = C_M \bar{P} \tag{2.17}$$

where $K \in \mathbb{R}^{3\times3}$ is the intrinsic parameters, $\begin{bmatrix} R & T \end{bmatrix} \in \mathbb{R}^{3\times4}$ represents the extrinsic parameters. $\bar{P}$ is the augmented vector of 3D coordinates and C_M is known as the camera matrix, $\tilde{p}$ is the homogeneous representation of a point in the image plane.

The calibration algorithm computes the camera matrix using the extrinsic and intrinsic parameters. The extrinsic parameters represent the location of the camera in the 3D scene. The intrinsic parameters depict the optical center and focal length of the camera, and these parameters also detail a projective transformation from 3D camera's coordinates into the 2D image coordinate. The relation between the extrinsic and intrinsic parameters is depicted in Fig. 2.5.

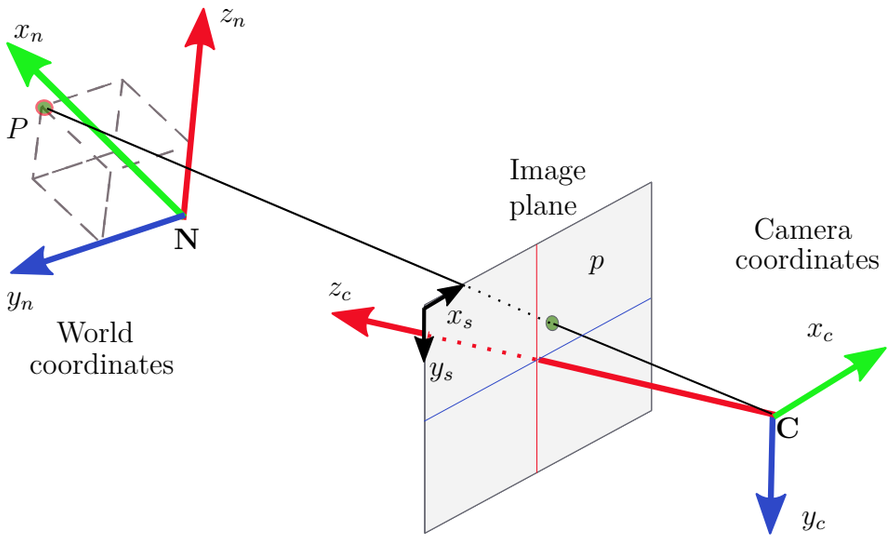


Figure 2.4 – The image plane using the image inverted to describe the projection of a point in the space 3D.

Intrinsic parameters

These parameters are given by the next expression:

$$\lambda \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = K\Pi_0\bar{P} = \begin{bmatrix} fs_x & fs_\theta & o_x \\ 0 & fs_y & o_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x_c \\ y_c \\ z_c \\ 1 \end{bmatrix} \tag{2.18}$$

where λ is a multiplicative scaling factor obtained by normalizing to 1 the last coordinate of the transformation result. The matrix Π_0 of size 3×4 represents the perspective projection. The triangular matrix K of size 3×3, brings together all the intrinsic parameters of a particular camera, which is named matrix of intrinsic parameters or camera's calibration matrix and contains the following geometric interpretations:

- o_x is the x coordinate of the main point in pixels
- o_y shows the y coordinate of the main point in pixels
- fs_x represents the size of pixels of unit length on the horizontal axis
- fs_y describes the size of pixels of unit length on the vertical axis
- fs_θ depicts the deviation of the pixel, often close to zero.

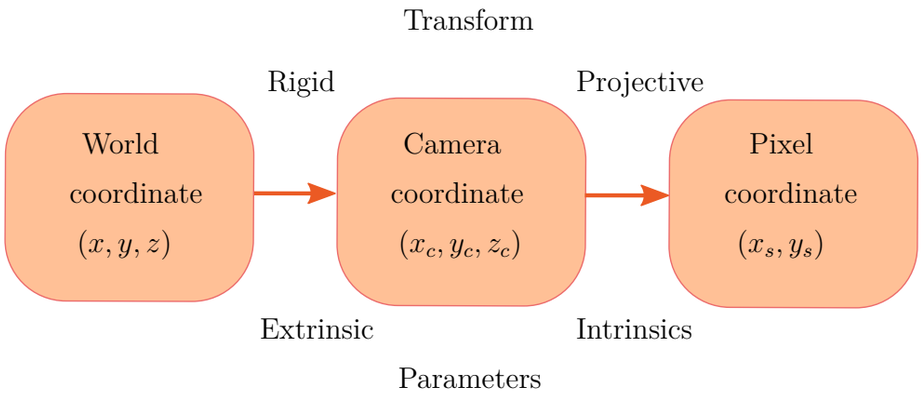


Figure 2.5 – Relation between extrinsic and intrinsic parameters.

Extrinsic parameters

The geometric relation between a given point of coordinates $\bar{P} = [x_n, y_n, z_n, 1]^T$ with respect to the world reference and its corresponding image, $\bar{x}' = [x', y', 1]^T$, depends upon the rigid body movement $[RT]$ between the world reference frame and camera reference, named extrinsic parameters.

Thus, the model of image formation is as follows:

$$\lambda \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & s_\theta & o_x \\ 0 & s_y & o_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} R_{3\times3} & T_{3\times1} \\ 0_{1\times3} & 1 \end{bmatrix} \begin{bmatrix} x_n \\ y_n \\ z_n \\ 1 \end{bmatrix} \tag{2.19}$$

which is the same equation given by (2.17).

2.1.4 Perspective projections

A perspective projection is distinguished from a parallel projection by the convergence of parallel lines, diminution of size, and nonuniform foreshortening. It means that parallel lines that are not parallel to the projection plane converge to a single point, called a vanishing point (vp). Vanishing points for lines parallel to a plane always lie along a straight line in the projection plane. When this line appears horizontal to the observer it is referred to as the horizontal line.

One of the distinguishing features of perspective projection is that the image of an object that stretches off to infinity can have finite extent. For example, an infinite scene line is imaged as a line terminating in a vanishing point. Similarly, parallel world lines, such as railway lines, are imaged as converging lines, and theirs intersection is the vanishing point for the direction of the railway. Thus, if we see the image perspective acquired from

a cube, see Fig. 2.7, perspective projections can be classified by the number of points perspective projection or finite vanishing points.

2.1.5 Vanishing point

Geometrically the vanishing point of a line is obtained by intersecting the image plane with a ray parallel to the world line and passing through the camera center. Thus a vanishing point depends only on the direction of a line, not on its position. Consequently a set of parallel world lines have a common vanishing point, as illustrated in the Fig. 2.8, Fig. 2.9 and Fig. 2.10.

Therefore, in general vanishing points have the next properties

- Any two parallel lines have the same vanishing point
- The ray from **C** through *vp* point is parallel to the lines
- An image may have more than one vanishing point

Fig. 2.6 shows graphically these properties.

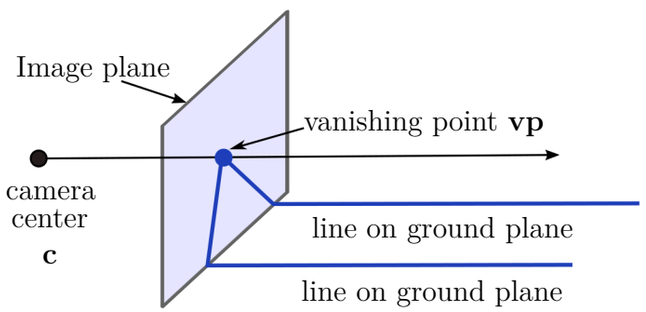


Figure 2.6 – Lines parallels in the space intersect in a vanishing point in the image plane.

One finite vanishing point

One-point perspective projection is the type of projection in which the projection plane intersects only one of the principal coordinate axes (one finite vanishing point). Hence, to obtain a one-point perspective, the projection plane must be parallel to one of the principal planes. One-point perspective of the cube is illustrated in Fig. 2.8 .

Two finite vanishing points

A two-point (or angular) perspective projection is the type of projection where the projection plane intersects two of the principal coordinate axes (two finite vanishing points).

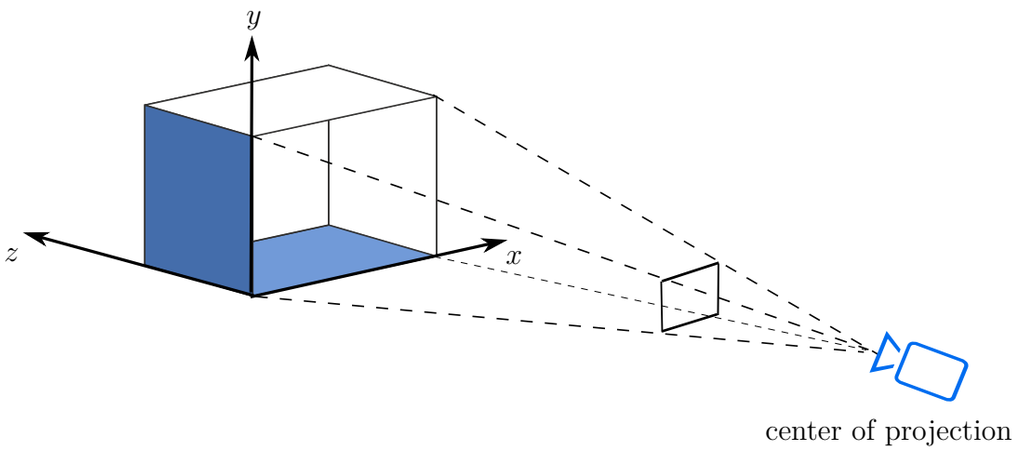


Figure 2.7 – Perspective projection of a cube.

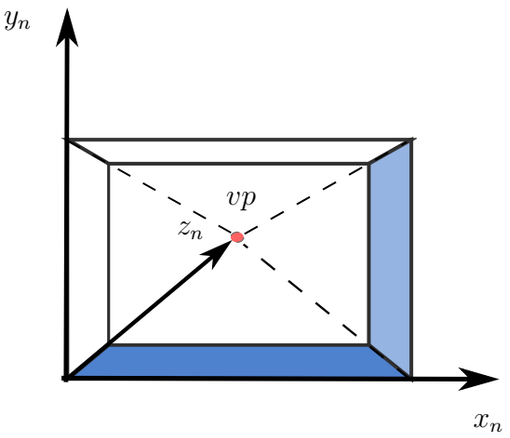


Figure 2.8 – One-point perspective projection resulting from the construction of one *vp*.

A two-point perspective is obtained by choosing the projection plane parallel to one of the principal axes, but not parallel to any coordinate plane. This perspective is illustrated seeing a cube in Fig. 2.9.

Three finite vanishing points

A three-point perspective projection is the type of projection where the projection plane intersects all coordinate axes (three finite vanishing points). A three-point perspective is obtained by choosing the projection plane that it is not parallel to any coordinate axis.

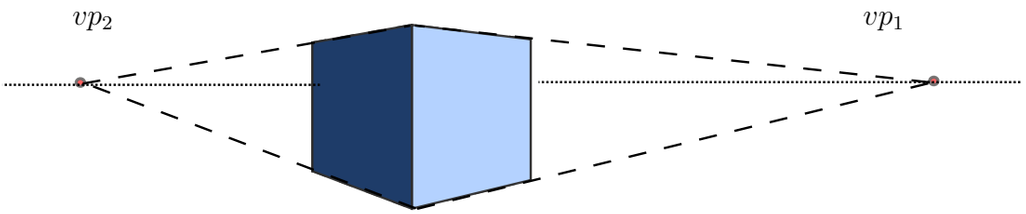


Figure 2.9 – Two-points perspective projection from the construction of two *vp*.

A three point perspective of a cube is illustrated in Fig. 2.10.

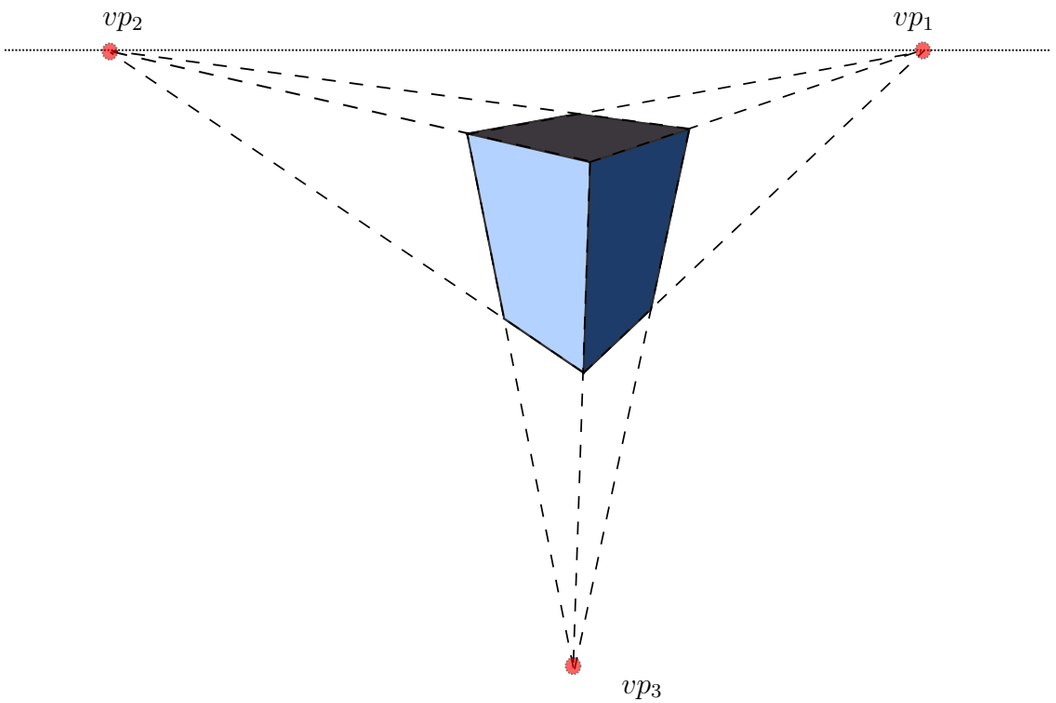


Figure 2.10 – Three-points perspective projection from the construction of three *vp*s.

2.2 Vanishing point extraction

Often vanishing points are computed from the image of a set of parallel line segments, though they may be determined in other ways for example by using equal length intervals on a line. In the case of the detected segments of parallels lines in the image plane, the objective is to estimate its common intersection, which is the image of the direction of

the parallel scene lines. Due to measurement noise and discretization the parallels lines will be not generally intersected in a unique point. Commonly the vanishing point is then computed by intersecting the lines pairwise and using the centroid of these intersections, or finding the closest point to all the measured lines.

There are several works on detecting vanishing points for different applications whose techniques can be divided into three categories. The first two require the knowledge of the internal parameters of the camera and the last one operates in an uncalibrated setting.

In the first one it is used a set of unit vectors on the Gaussian sphere centered on the optical center of the camera Collins and Weiss [1990], Boulanger et al. [2006a]. Approach proposed by Barnard [1983] was performed on a quantized Gaussian sphere using a Hough transform, until this was shown to lead to spurious vanishing points. Since then, most of the works rely on 3D parallelism or orthogonality of dominant structures in the scene to avoid any false detection.

More recent techniques, requiring the knowledge of internal parameters, are based on the "Manhattan assumptions" that the prominent structures of the scene are orthogonal to each other Coughlan and Yuille [1999]. The algorithms directly estimate the so-called Manhattan directions or equivalently the camera orientation Boulanger et al. [2006a], Elloumi et al. [2012]. Our work uses this technique due to quadrotor environment whose geometry its similar to the proposed approach of this method.

Finally, some algorithms assume no knowledge of the internal parameters and their main goal is to estimate possibly non-orthogonal vanishing points. These are especially useful for calibration of the camera using one or more views, but also to recover building facades. The Expectation Maximization (EM) approach of Antone and Teller [2000] requiring entities represented on Gaussian sphere. The EM approach requires an initial estimate of the vanishing points.

2.3 Summary

The preliminaries and concepts from computer vision were described in this chapter. Camera parameters were also introduced to see the relation between a point in the real world and a point in the image plane. Then, perspective projection was detailed in order to see how vanishing points can describe the environment. Finally, a brief survey of vanishing point was presented.

Chapter 3
Preliminaries UAVs

This chapter is focused on describing some characteristics and operation of a quadrotor vehicle. First, Linear representation between body reference frame and world reference frame is presented in section 3.2. Then, a survey of some angular representation, which are Euler, Cardan angles and quaternions, is detailed in section 3.3. Next, in section 3.4 is shown a 3D pose taking into account position and rotation given in precedent sections. After, rigid body representation is detailed in section 3.1. Then, an introduction and a mathematical model of the quadrotor system are given in section 3.5.1. Finally, a relation between coordinate frames (world, camera and body) is examined in section 3.6.

3.1 Rigid body representation

A rigid body representation is usually represented by the body reference frame $\mathbf{B}[\vec{e}_1^b, \vec{e}_2^b, \vec{e}_3^b]$ which is located at the center of mass of the rigid body, and the world reference frame $\mathbf{N}[\vec{e}_1^n \vec{e}_2^n, \vec{e}_3^n]$ which is located at some point in the space, see Fig. 3.4 (the earth NED frame is normally used for a quadrotor system). The body attitude in the space can be represented in many ways, each one with their advantages and disadvantages, depending mainly on the targeted application. For quadrotor vehicles, the body axes x_b, y_b and z_b coincide with the principal axes of inertial and North/West orientation (or room main axes for indoor navigation).

The motion of the body frame is described relative to the world reference frame. Usually, the position and orientation of the vehicle is described relative to the world reference frame while the linear and angular velocities of the vehicle is expressed in the body reference frame.

Particularly, the general motion of rigid body can be described by the following vectors:

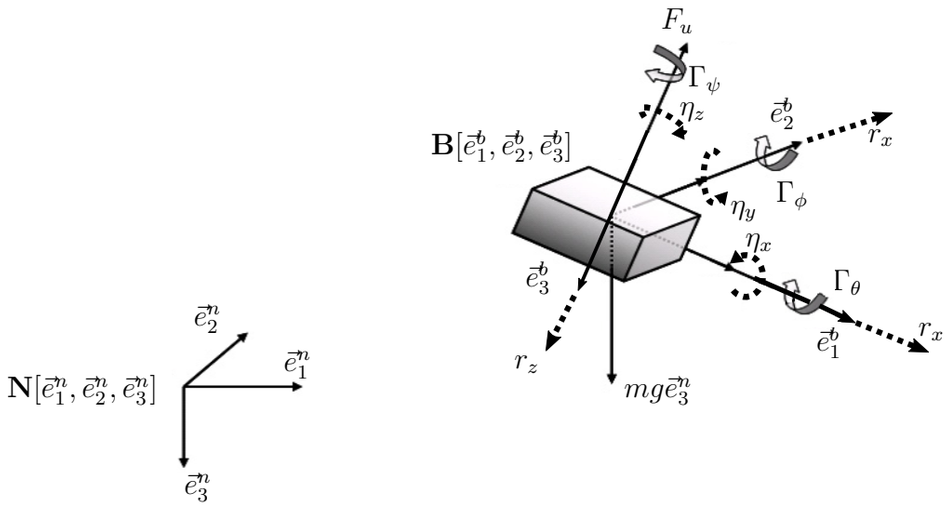


Figure 3.1 – Rigid body representation.

$$\begin{aligned} \vec{\xi} &= [x_n, y_n, z_n]^T; \quad & \vec{E} &= [\theta, \phi, \psi]^T; \\ \vec{r} &= [r_x, r_y, r_z]^T; \quad & \vec{\eta} &= [\eta_x, \eta_y, \eta_z]^T; \\ \vec{F} &= [F_x, F_y, F_z]^T; \quad & \vec{\Gamma} &= [\Gamma_\phi, \Gamma_\theta, \Gamma_\psi]^T \end{aligned} \tag{3.1}$$

Here $\vec{\xi}$ and $\vec{E}$ denotes position and orientation respectively. $\vec{r}$ is the linear velocity. $\vec{\eta}$ represents the angular velocity. $\vec{F}$ and $\vec{\Gamma}$ are the forces and moments acting in the body frame. $\vec{E}$ is given by the angles pitch, roll and yaw angles (θ, ϕ, ψ); however, a representation in quaternions q was also proposed above.

3.2 Linear representation

The 3D space is typically represented by three axes denoted z-axis, that is orthogonal to both the x- and y-axes. The direction of the z-axis obeys the right-hand rule and forms a right-handed coordinate frame with the x- and y-axes. Fig. 3.2 shows a body coordinate frame $\mathbf{B}$ that we wish to describe with respect to the world coordinate frame $\mathbf{N}$. Unit vectors parallel to the axes are denoted by $\mathbf{N}[\vec{e}_1^n, \vec{e}_2^n, \vec{e}_3^n]$ and $\mathbf{B}[\vec{e}_1^b, \vec{e}_2^b, \vec{e}_3^b]$ which represent the orthonormal directions of the world reference frame and body coordinate frame respectively. A point P is represented by its $x-, y-$ and z-coordinates $[x, y, z]^T$ or as a bound vector. For instance, in Fig. 3.2 a point P with respect to the world reference frame can be given as ${}^{\mathbf{N}}P = [x_n, y_n, z_n]^T$ or $P = x_n\vec{e}_1^n + y_n\vec{e}_2^n + z_n\vec{e}_3^n$.

Fig. 3.2 also shows that the origin of $\mathbf{B}$ has been displaced by the vector ${}^{\mathbf{N}}T_{\mathbf{B}} =$

$[x_n, y_n, z_n]^T$ and then rotated in some complex fashion with respect to the frame **N**. The leading superscript denotes the reference coordinate frame and the subscript denotes the frame being described. If the initial superscript is missing we assume that the change in pose is relative to the world coordinate frame which is generally denoted as 0. ${}^{\mathbf{N}}T_{\mathbf{B}}$ are the coordinates of the origin of **B** in the frame **N**. This approach is considering an arbitrary point P with respect to each of the coordinate frames and to determine the relationship between ${}^{\mathbf{N}}P$ and ${}^{\mathbf{B}}P$. In addition, it is shown linear and angular velocities $(\vec{r}, \vec{\eta})$, which represent the motions in time of the body reference frame **B**. We will again consider the problem in two parts: rotation and then translation. The translational velocity is direct: it is the rate of change of the position of the origin of the coordinate frame. Rotation and angular rotation are surprisingly complex for the 3-dimensional case and we devote all of the next section to it.

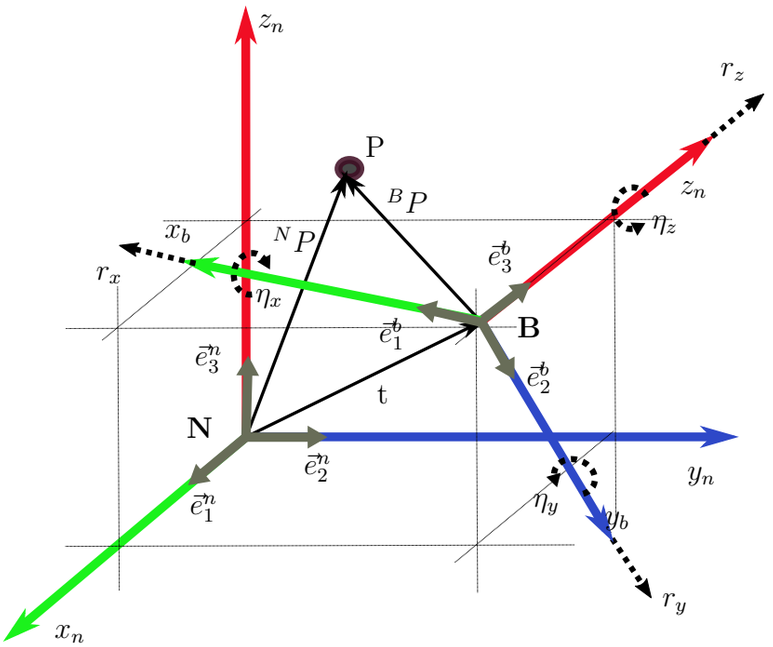


Figure 3.2 – Two coordinate frames, a point p whose coordinates can be given with respect to frame **B** or **N**.

3.3 Angular representation

Mathematically, a rotation is a rigid body movement which, unlike a translation, keeps a point fixed. This definition applies to rotations within both two and three dimensions (in a plane and in space, respectively). Rotations around the x, y and z axes are called

principal rotations. Rotation can be performed by taking into account a combination of rotations around each axis. That is to say, any spatial rotation can be decomposed into a combination of principal rotations.

3.3.1 Orthonormal Rotation Matrix

The orientation of a coordinate frame can be represented by their unitary vectors expressed in terms of the reference coordinate frame. Each unitary vector has three elements and they form the columns of a 3×3 orthonormal matrix R

$$\begin{bmatrix} x_n \\ y_n \\ z_n \end{bmatrix} = R \begin{bmatrix} x_b \\ y_b \\ z_b \end{bmatrix} \tag{3.2}$$

which transforms the description of a vector defined with respect to frame **B** to a vector with respect to **N**.

Principal rotations

The orthonormal rotation matrices for rotation of ϕ, θ and ψ about the x-, y- and z-axes are respectively in (3.3). This yields the following transformations matrices:

$$R_{x,\phi} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & c_\phi & s_\phi \\ 0 & -s_\phi & c_\phi \end{bmatrix} \quad R_{y,\theta} = \begin{bmatrix} c_\theta & 0 & -s_\theta \\ 0 & 1 & 0 \\ s_\theta & 0 & c_\theta \end{bmatrix} \quad R_{z,\psi} = \begin{bmatrix} c_\psi & s_\psi & 0 \\ -s_\psi & c_\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.3}$$

where $s_\cdot = \sin(\cdot)$ and $c_\cdot = \cos(\cdot)$. $R_{x,\phi}$, $R_{y,\theta}$ and $R_{z,\psi}$ denote a rotation for a given angle about each axis.

Coordinate transformation matrix

Rotation matrix R belongs to the subspace of orthogonal matrices of dimension three, called special orthogonal group, denoted by $S0(3)$, and defined by

$$S0(3) = \{R | R \in \mathbb{R}^{3\times 3}, \quad R^T R = I_3, \quad \det(R) = 1\} \tag{3.4}$$

In a rotation matrix R, each element R_{ij} is a direct cosine, given by:

$$R = \begin{bmatrix} R_{11} & R_{12} & R_{13} \\ R_{21} & R_{22} & R_{23} \\ R_{31} & R_{32} & R_{33} \end{bmatrix} \tag{3.5}$$

where

$$\vec{R_1} = \begin{bmatrix} R_{11} \\ R_{21} \\ R_{31} \end{bmatrix} \quad \vec{R_2} = \begin{bmatrix} R_{12} \\ R_{22} \\ R_{32} \end{bmatrix} \quad \vec{R_3} = \begin{bmatrix} R_{13} \\ R_{23} \\ R_{33} \end{bmatrix} \tag{3.6}$$

consequently,

$$R = \begin{bmatrix} \vec{R_1} & \vec{R_2} & \vec{R_3} \end{bmatrix} \tag{3.7}$$

where

$$R_i^T R_i = 1 \quad \text{and} \quad R_i^T R_j = 0 \quad \forall i \neq j \tag{3.8}$$

Three- Angles Representation

Euler's rotation theorem requires successive rotation about three axes such that no two successive rotations are about the same axis. There are two classes of rotation sequence: Eulerian and Cardanian.

The Eulerian type involves repetition, but not successive, of rotations about one particular axis: XYX, XZX, YXY, YZY, ZXZ, or ZYZ. The Cardanian type is characterized by rotations about all three axes: XYZ, XZY, YZX, YXZ, ZXY, or ZYX. The ZYZ sequence is commonly used in aeronautics and mechanical dynamics. The Euler angles are the 3-vector $[\phi, \theta, \psi]$.

Another widely used convention are the Cardan angles: roll, pitch and yaw. In literature, roll-pitch-yaw sequence is usually defined as ZYX or XYZ depending on whether they have mobile robot or robot arm focused. When describing the attitude of vehicles such as ships, aircrafts and cars, the convention is that the x-axis points in the forward direction and the z-axis points either up or down. This leads to the ZYX angle sequence.

$$^{\mathbf{N}}R_{\mathbf{B}} = R = R_{z,\psi}^T R_{y,\theta}^T R_{x,\phi}^T = \begin{bmatrix} c_\theta c_\psi & s_\phi s_\theta c_\psi - c_\phi s_\psi & c_\phi s_\theta c_\psi + s_\phi s_\psi \\ c_\theta s_\psi & s_\phi s_\theta s_\psi + c_\phi c_\psi & c_\phi s_\theta s_\psi - s_\phi c_\psi \\ -s_\theta & s_\phi c_\theta & c_\phi c_\theta \end{bmatrix} \tag{3.9}$$

The roll-pitch-yaw sequence allows all angles to have arbitrary sign and it has a singularity when $\theta = \pm\frac{\pi}{2}$ which is fortunately outside the range of feasible attitudes for most vehicles.

Rotation about an Arbitrary Vector

Two coordinate frames of arbitrary orientation are related by a single rotation about some axis in space defined by its rotation matrix R. From the definition of eigenvalues and eigenvectors we recall that

$$R\vec{a} = \lambda\vec{a} \tag{3.10}$$

where $\vec{a}$ is the eigenvector corresponding to the eigenvalue λ. For the case $\lambda = 1$ verified by rotation matrices:

$$R\vec{a} = \vec{a} \tag{3.11}$$

which implies that the corresponding eigenvector $\vec{a}$ is unchanged by the rotation R. For any real factor r, $r\vec{a}$ will remained unchanged by the rotation R. An orthonormal rotation matrix will always have one real eigenvalue at $\lambda = 1$ and in general a complex pair $\lambda = \cos\beta \pm i\sin\beta$ where β is the rotation angle. Hence, once R is known, the rotation axis $\vec{a}$ and the rotation angle β can be easily obtained. The inverse problem, that is converting from angle and vector to a rotation matrix, can be achieved using Rodrigues' rotation formula

$$R = I_{3\times 3} + \sin\beta[\vec{a}\,]_\times + (1-\cos\beta)[\vec{a}\,]_\times^2 \tag{3.12}$$

where I_3 represents the identity matrix of dimension three and $[\vec{a}\,]_\times$ represents the skew-symmetric matrix, given by:

$$[\vec{a}\,]_\times = \begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix}_\times = \begin{bmatrix} 0 & -a_z & a_y \\ a_z & 0 & -a_x \\ -a_y & a_x & 0 \end{bmatrix} \tag{3.13}$$

It is interesting to note that this representation of an arbitrary rotation is parameterized by four numbers: three for the rotation axis, and one for the angle of rotation. However, the direction can be represented by a unitary vector $\vec{b} = \vec{a}/||\vec{a}||$ which has only two parameters and the angle can be encoded in the length to give a 3-parameter representation such as $\vec{b}\beta$, $\vec{b}\sin(\beta/2)$, $\vec{b}\tan(\beta)$ or the Rodrigues' vector $\vec{b}\tan(\beta)$. While these forms are minimal and efficient in terms of data storage they are analytically problematic and ill-defined when $\beta = 0$.

Matrix Exponentials

The exponential map effects a transformation from the axis-angle representation of rotations to rotation matrices. Essentially, by using a Taylor expansion one derives a closed-form relation between these two representations. Given a unitary vector $\vec{a}$ representing the unitary rotation axis, and an angle, β, an equivalent rotation matrix R is given as follows,

$$R = e^{[\vec{a}]_\times \beta} \in \mathbb{R}^3 \tag{3.14}$$

the notation $[\cdot]_\times : \mathbb{R}^3 \mapsto \mathbb{R}^3 \times \mathbb{R}^3$ indicates the mapping from a vector to a skew-symmetric matrix (3.13). Since $[\vec{a}]_\times \beta = [\vec{a}\beta]_\times$ we can treat $\vec{a}\beta \in \mathbb{R}^3$ as rotational parameters called

exponential coordinates.

Rotational velocity

The body-fixed angular velocity vector $\vec{\eta} = [\eta_x, \eta_y, \eta_z]^T$, see Fig. 3.2, and the Euler rate vector $\dot{\vec{W}} = [\dot{\phi}, \dot{\theta}, \dot{\psi}]^T$ are related through a transformation matrix $W(\vec{E})$, sometimes called the Wronskian matrix, according to:

$$\dot{\vec{E}} = W(\vec{E})\vec{\eta} \tag{3.15}$$

The orientation of a body reference frame **B** w.r.t. the inertial reference frame **N** is given by

$$\vec{\eta} = \begin{bmatrix} \dot{\phi} \\ 0 \\ 0 \end{bmatrix} + R_{x,\phi} \begin{bmatrix} 0 \\ \dot{\theta} \\ 0 \end{bmatrix} + R_{x,\phi} R_{y,\theta} \begin{bmatrix} 0 \\ 0 \\ \dot{\psi} \end{bmatrix} = W^{-1}(\vec{E})\dot{\vec{E}} \tag{3.16}$$

Then, the kinematic equation is given by Fossen [1994]:

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} 1 & \tan\theta \sin\phi & \tan\theta \cos\phi \\ 0 & \cos\phi & -\sin\phi \\ 0 & \frac{\sin\phi}{\cos\theta} & \frac{\cos\phi}{\cos\theta} \end{bmatrix} \begin{bmatrix} \eta_x \\ \eta_y \\ \eta_z \end{bmatrix} \tag{3.17}$$

3.3.2 Attitude error

Two orientations of a rigid body are considered, described by rotation matrices R_1 and R_2 respectively. Then, the relative attitude between these two orientations is computed by:

$$R_r = R_1^{-1} R_2 \tag{3.18}$$

In fact, R_r represents an operator of orientation which rotates R_2 about R_1. From here, the relative orientation is used in the estimation and in the orientation control as *attitude error*. If $R_d = R_1$ is the desired attitude of a rigid body and $R = R_2$ is the real attitude of the body, then the attitude error is computed by:

$$R_e = R_d^{-1} R \tag{3.19}$$

If the attitude error is zero, then $R_e = I_3$. In terms of quaternions.

3.4 Pose in 3D

The position and orientation change between the two coordinate frames as shown in Fig. 3.3. This is often referred to as a rigid-body displacement or rigid-body motion. Above, several different representations of orientation were discussed, and we need to combine one of these with translation, to create a tangible representation of relative pose. The simplest way to represent the 3D pose is as follow:

$$^{N}P = R^{B}P +^{\mathbf{N}} T_{\mathbf{B}}. \tag{3.20}$$

Homogeneous Transformation Matrix

Considering the translation between the origins of the frames shown in Fig. 3.3. $^{\mathbf{N}}\tilde{P} = [x_n, y_n, z_n, 1]^T$ and $^{\mathbf{B}}\tilde{P} = [x_b, y_b, z_b, 1]^T$ are the homogeneous representations of $^{\mathbf{N}}P = [x_n, y_n, z_n]^T$ and $^{\mathbf{B}}P = [x_b, y_b, z_b]^T$ respectively. The tilde indicates the vector is homogeneous. The relation becomes as follow:

$$^{\mathbf{N}}\tilde{P} = \begin{bmatrix} R & ^{\mathbf{N}}T_{\mathbf{B}} \\ 0_{1x3} & 1 \end{bmatrix}^{\mathbf{B}} \tilde{P} \tag{3.21}$$

$^{\mathbf{N}}T_{\mathbf{B}}$ is the vector defining the origin of frame $\mathbf{B}$ with respect to frame $\mathbf{N}$, see section 3.2, and R is the 3×3 orthonormal matrix which describes the orientation of the axes of frame $\mathbf{B}$ with respect to frame $\mathbf{N}$. If a point is represented by its homogeneous coordinates then

$$^{\mathbf{N}}\tilde{P} = {}^{\mathbf{N}}M_{\mathbf{B}}\ {}^{\mathbf{B}}\tilde{P} \tag{3.22}$$

and $^{\mathbf{N}}M_{\mathbf{B}}$ is a 4×4 homogeneous transformation matrix. The 4×4 homogeneous transformation is very commonly used in robotics, computer graphics and computer vision, see Chapter 2.

3.5 Quadrotor mathematical model

The dynamic model of the quadrotor vehicle can be divided into two parts: attitude and translation. The attitude system is covered by the rotational part, and the position system is described by the translational part. There are different approaches to describe the dynamics of the system. For example, the Newton-Euler approach which takes into account the concepts of forces and torques. There is also the Euler-Lagrange approach which considers the concepts of kinetic and potential energy. In this work, the unitary quaternion to represent the attitude of the system and the Newton-Euler approach are

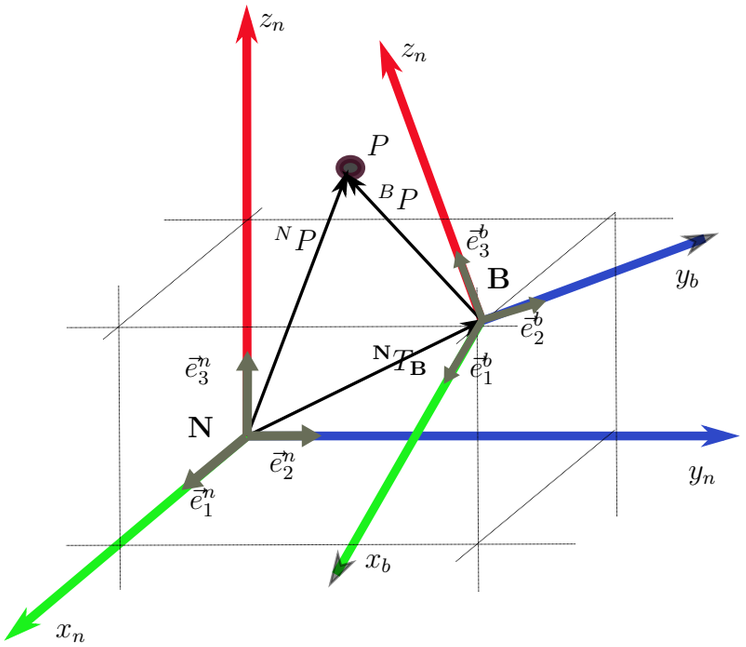


Figure 3.3 – Two coordinate frames, a point p whose coordinates can be given with respect to frame $\mathbf{B}$ or $\mathbf{N}$.

used in order to obtain the dynamical equations of the system. In order to obtain the equations of motion of the aerial system, we make the following assumptions:

- the cross-shaped structure is supposed to be rigid,
- the quadrotor has a perfectly symmetrical structure, which allows to consider a diagonal inertia matrix,
- the pushing force f_i and the reactive torque Q_i produced by each rotor are supposed proportional to the square of the speed of rotation of the blade,
- the vehicle flight is assumed to be performed under conditions of a standard atmosphere,
- the disturbances produced by the air are neglected.

The modelling of the system has been reviewed in many works, and is given by (3.23) and (3.24). The six degrees of freedom (position and attitude) of the system could be separated into translational (Σ_T) and rotational (Σ_R) motions, as follows

$$\Sigma_T \quad : \quad \begin{cases} \dot{\vec{\xi}} = \vec{r} \\ m\dot{\vec{r}} = mg\vec{e}_3^n - F_u R^T \vec{e}_3^n \end{cases} \tag{3.23}$$

$$\Sigma_R \quad : \quad \begin{cases} \dot{q} = \dfrac{1}{2}\Xi(q)\ \vec{\eta} \\ J\dot{\vec{\eta}} = -[\vec{\eta}\,]_\times J\vec{\eta} + \vec{\Gamma} + \vec{\Gamma}_G \end{cases} \tag{3.24}$$

where m denotes the mass of the vehicle, J its inertial matrix expressed in $\mathbf{B}$. R is the rotation matrix which allows the passage between the inertial and fixed frames. F_u represents the total thrust. g indicates the gravity acceleration. $\vec{e}_3^n = [0\ 0\ 1]^T$ is the direction orthonormal of z_n-axis of world reference frame. $\xi = [x_n, y_n, z_n]^T$ represents the position of the vehicle center of gravity, which coincides with the origin of frame $\mathbf{B}$, with respect to frame $\mathbf{N}$, $\vec{r} = [r_x, r_y, r_z]^T$ its linear velocity in $\mathbf{N}$, q is the orientation quaternion and $\vec{\eta}$ denotes the angular velocity of the vehicle expressed in $\mathbf{B}$. $\vec{\Gamma}$ is the vector of control torques, and $\vec{\Gamma}_G$ represents the gyroscopic torques. Note that the rotation matrix R can be given as a function of Euler angles. Fig. 3.4 depicts the basic representation of a quadrotor system which replace the basic representation of a rigid body shown in Fig. 3.1.

3.5.1 Characteristics and operation of the quadrotor

The quadrotor or four rotor helicopter is a mechatronic system composed classically of a cross structure. At each end of the cross there is a propeller coupled to a motor, and at the center of the configuration all the electronic elements are established (power source, computer, etc). Compared to the classical helicopter, this system does not have main rotor and the control is performed by the angular velocity changes on each rotor, (Nelson [1998]).

The four rotors are composed of the propellers coupled to DC motors or DC brushless motors (BLDC). The quadrotor prototype is represented in Fig. 3.4, where the front and rear motors (1 and 2) rotate clockwise, while the others two (3 and 4) rotate counter-clockwise. In this way, gyroscopic effects and aerodynamic torques tend to cancel each other out in trimmed flight.

Each rotor produces a force f_i parallel to its rotation axis, as well as a drag torque Q_i, opposite to the direction of rotation. The total force or total thrust acting on the aircraft (parallel to the z_b axis) is the sum of the four forces generated by each rotor ($F_u = f_1 + f_2 + f_3 + f_4$). The combination of these forces and the drag torques allow the angular motions over the main axes of the vehicle, see Fig. 3.5.

- **Roll** (ϕ): It is produced by the difference $f_3 - f_4$. To obtain this, the velocity of

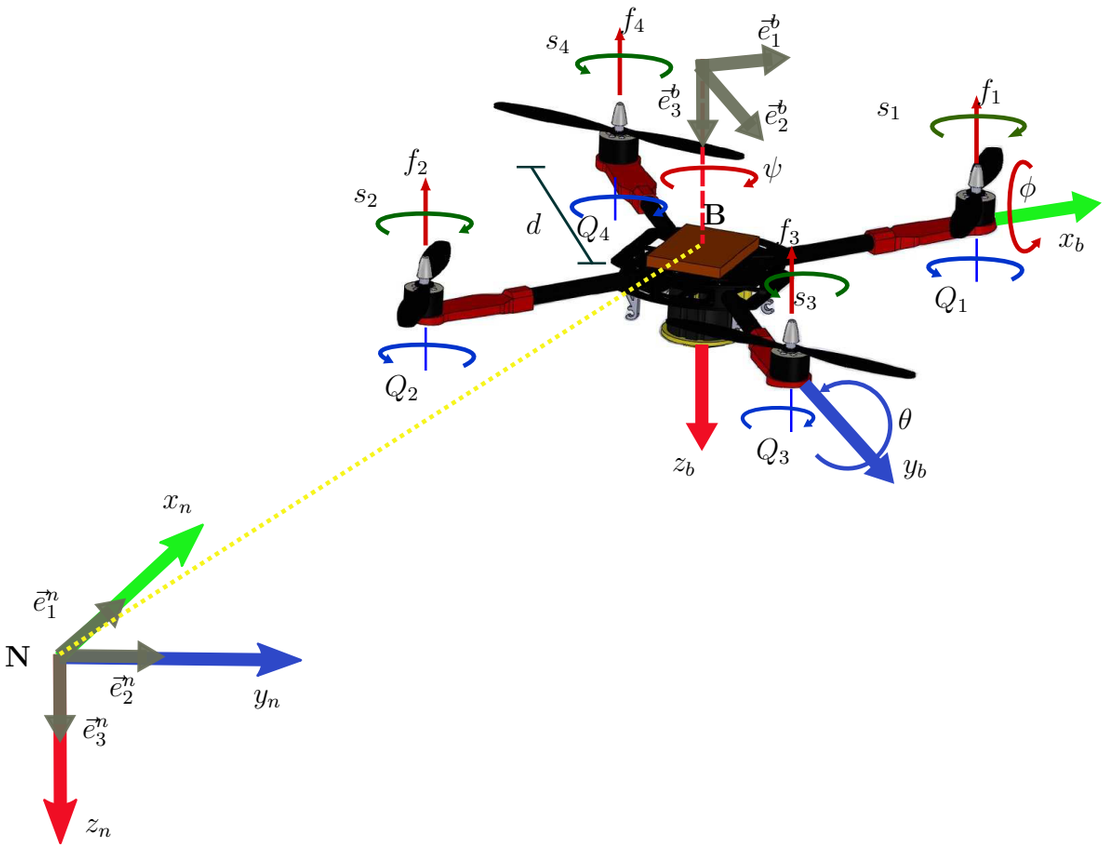


Figure 3.4 – Scheme of the quadrotor configuration: inertial frame $\mathbf{N}[\vec{e}_1^n \vec{e}_2^n, \vec{e}_3^n]$, the body fixed frame $\mathbf{B}[\vec{e}_1^b, \vec{e}_2^b, \vec{e}_3^b]$, the f_i force on each motor, angular velocity of the motors s_i and the reaction torques Q_i.

the motor m_3 is increased/reduced, while the velocity of the motor m_4 is equally decreased/increased. This difference of forces produces a torque Γ_ϕ around the axis x_b.

- **Pitch** (θ): It is produced by the difference $f_1 - f_2$. It is obtained similarly using the front and rear motors m_1 and m_2. This difference of forces produces a torque Γ_θ around the axis y_b.

- **Yaw** (ψ): It is the combination of all the reactive torques, $Q_1 + Q_2 - Q_3 - Q_4$. It is obtained by decreasing/increasing the speed of the front and rear motors while decreasing/increasing the speed of the lateral motors. In other words, if there is a difference of speed between the motors turning in the opposite direction, the reactive torques produce a torque Γ_ψ around the axis z_b.

- **Vertical displacement on the x_n axis**: To go forward or back, the rotational

speed of motor m_2 must be decreased/increased, while decreasing/increasing the rotational speed of the motor m_1.

- **Lateral displacement on the y_n axis**: To go to the right or left, the rotational speed of the lateral motors m_4 and m_3 must be decreased/increased.

- **Displacement on the z_n axis**: To go up or down, the forces of all the rotors m_i must be decreased/increased. In the absence of disturbances, the aerial system can perform a hover at a certain height by having a zero translation speed. Then, the total thrust F_u must balance the weight mg of the aerial system by pointing its direction in the axes z_b.

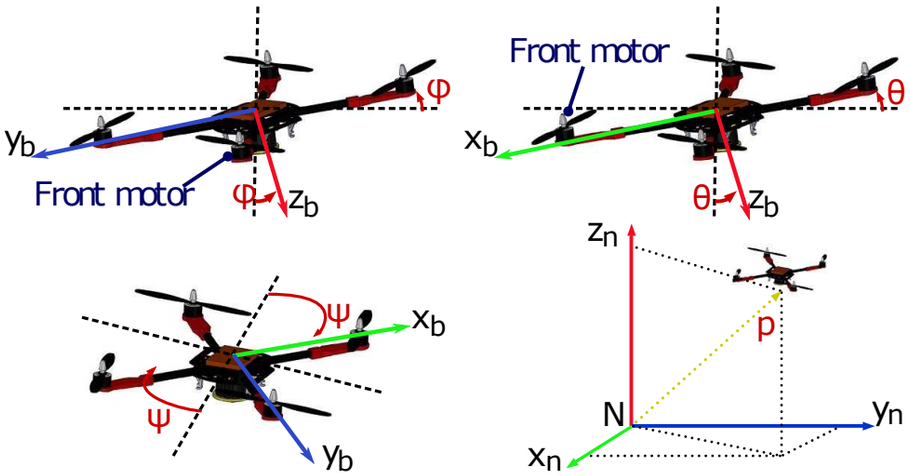


Figure 3.5 – Roll (ϕ), pitch (θ), yaw (ψ) and space displacement.

From the description of the different angular movements and the vertical and horizontal displacements, it is visible that the position of the system depends on its attitude.

Thrust force. According to Fig. 3.4 the force f_i delivered by the i^{th} rotor with a blade rotational speed s_i is modeled by:

$$f_i = b_F s_i^2 \quad \text{with} \quad i \in 1, 2, 3, 4 \tag{3.25}$$

where $b_F > 0$ is the thrust force parameter. The expression F_u is given by:

$$F_u = \sum_{i=1}^{4} f_i = b_F \sum_{i=1}^{4} s_i^2. \tag{3.26}$$

Reactive force. The reactive torque Q_i generated in the free air by the rotor i due to the motor drag and the total thrust F_u produced by the four rotors can be, respectively,

approximated by Castillo et al. [2004]

$$Q_i \approx k_Q s_i^2 \tag{3.27}$$

where $k > 0$ is the motor drag constant. Both b_F and k_Q are parameters that depend on the density of air, the radius, the shape, the pitch angle of the blade and other factors.

Gyroscopic torques. The vector of gyroscopic torques Γ_G is a consequence of the simultaneous rotation of the structure of the quadrotor and the high-speed rotation of the actuators, and it is given by:

$$\Gamma_G = \sum_{i=1}^{4} J_r(\vec{w} \times \vec{z}_b)(-1)^{i+1} s_i \tag{3.28}$$

where J_r is the inertia of the so-called rotor (composed of the motor rotor itself with the gears). This vector adds a term that is canceled because of relation: $\vec{\eta}^T \Gamma_G = \vec{\eta}^T(\vec{\eta}^T \times \vec{e_3^b}) \sum_{i=1}^{4} J_r(-1)^{i+1} s_i = 0$ Guerrero-Castellanos et al. [2011a].

Control torques. As it can be seen above the components of the control torque $\vec{\Gamma} \in \mathbb{R}^3$ generated by the rotors are given by $\vec{\Gamma} = [\Gamma_\phi,\ \Gamma_\theta,\ \Gamma_\psi]^T$, with

$$\Gamma_\phi = d(f_3 - f_4) = db_F(s_3^2 - s_4^2) \tag{3.29}$$

$$\Gamma_\theta = d(f_1 - f_2) = db_F(s_1^2 - s_2^2) \tag{3.30}$$

$$\Gamma_\psi = Q_1 + Q_2 - Q_3 - Q_4 = k_Q(s_1^2 + s_2^2 - s_3^2 - s_4^2) \tag{3.31}$$

where d represents the distance from a rotor to the center of mass of the quadrotor. Combining the equation (3.26) with (3.29)-(3.31), the torques and forces applied to the vehicle are written in vector form as

$$\begin{bmatrix} \vec{\Gamma} \\ F_u \end{bmatrix} = \begin{bmatrix} 0 & 0 & db_F & -db_F \\ db_F & -db_F & 0 & 0 \\ k_Q & k_Q & -k_Q & -k_Q \\ b_F & b_F & b_F & b_F \end{bmatrix} \begin{bmatrix} s_1^2 \\ s_2^2 \\ s_3^2 \\ s_4^2 \end{bmatrix} = NS \tag{3.32}$$

with S the square rotor speeds of the four motors. It should be noted that the relations between the rotation velocities and the resistive torque of the actuator axis and those of the motor are given by

$$\begin{cases} z_m = K_g z \\ Q_m = \frac{Q}{K_g} \end{cases} \tag{3.33}$$

where z_m and Q_m represent the rotation speed and the resistive torque of the motor, respectively, and K_g defines the gear ratio. One of the main characteristics of the quadrotor is its symmetry. Consequently, the distribution of the mass can be considered uniform. Considering that the structure is similar at that one of the Fig. 3.4, the classical definitions of inertial moments and inertia product are used, see Fossen [1994].

3.6 Relation from world, camera and body frames to merge in to a complete system

In section 3.1, quadrotor vehicle was described by vectors depicting translation and rotational movements $(\vec{\xi}, \vec{E})$, linear and angular velocity $(\vec{r}, \vec{\eta})$, and finally force and torques acting on each axis $(\vec{F}, \vec{\Gamma})$. However, our system also takes into account an embedded camera which is put on the quadrotor system. This camera is used to take images from the environment. This information is used to estimate certain position of the system (quadrotor) with respect to its environment. The camera is put in front of the quadrotor, and the origin of camera reference frame $\mathbf{C}[\vec{e}_1^{\mathbf{c}}, \vec{e}_2^{\mathbf{c}}, \vec{e}_3^{\mathbf{c}}]$ is on the center of camera lens. The origin of body reference frame $\mathbf{B}$ is supposed to be in the center of gravity of the quadrotor system. Fig. 3.6 shows an general scheme of the system.

Once the references systems are defined, the relation between the world reference frame, camera reference frame, and body (quadrotor) reference frame can be obtained as a simple transformation. Fig. 3.6 depicts a point P which is projected into the image plane acquired by the embedded camera. The coordinates of the point P with respect to the camera coordinate system $\mathbf{C}$ can give us some useful information to know the position of the system. Fig. 3.6 also shows a ray which is going out from the camera projection center, and it goes directly to the point. Thus, point P can be expressed w.r.t. camera reference frame $\mathbf{C}$ as follows

$$^{\mathbf{C}}P = {}^{\mathbf{C}}R_{\mathbf{N}}{}^{\mathbf{N}}P + {}^{\mathbf{C}}T_{\mathbf{N}} \tag{3.34}$$

where $^{\mathbf{C}}P$ is the coordinate of the point P with respect to the camera $\mathbf{C}$. $^{\mathbf{N}}P$ represents the same point P with respect to the world $\mathbf{N}$. $^{\mathbf{C}}R_{\mathbf{N}}$ depicts the rotation from the world reference frame $\mathbf{N}$ w.r.t. the camera reference frame $\mathbf{C}$. Finally $^{\mathbf{C}}T_{\mathbf{N}}$ shows the translation from the world reference frame $\mathbf{N}$ w.r.t. the camera reference frame $\mathbf{C}$.

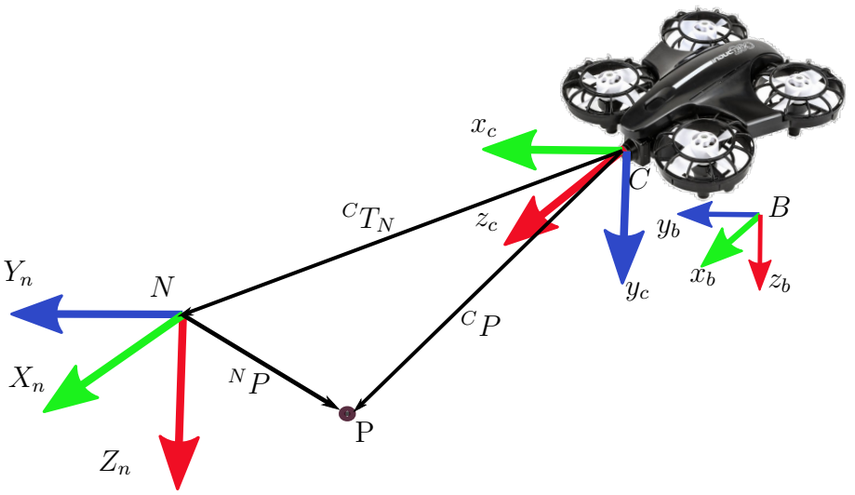


Figure 3.6 – The camera model system has its center of projection on the lens, which is the camera reference frame **C**. The z_c-axis is going outwards to the optical axis, x_c-axis to the right and the y_c-axis downwards.

3.6.1 Translation

The meaning of the translation ${}^{\mathbf{C}}T_{\mathbf{N}}$ between two references frames can be understood letting ${}^{\mathbf{N}}P$ to zero. Then, eq. (3.35) is the vector from the camera origin to world origin.

$$ {}^{\mathbf{C}}P = {}^{\mathbf{C}}R_{\mathbf{N}}(0) + {}^{\mathbf{C}}T_{\mathbf{N}} \tag{3.35} $$

3.6.2 Rotation

Let the rotation matrix be written as three orthogonal column vectors, see Fig. 3.7:

$$ {}^{\mathbf{C}}R_{\mathbf{N}} = \begin{bmatrix} {}^{\mathbf{C}}\vec{e}_1^{\,n} & {}^{\mathbf{C}}\vec{e}_2^{\,n} & {}^{\mathbf{C}}\vec{e}_3^{\,n} \end{bmatrix} \tag{3.36} $$

The rotation of the world **N** with respect to the camera **C** is shown in Fig. 3.7, and it is given by

$$ {}^{\mathbf{C}}R_{\mathbf{N}} = R_{\mathbf{z},\psi}R_{\mathbf{y},\theta}R_{\mathbf{x},\phi} \tag{3.37} $$

Taking into account the rotation matrix given in 3.3 and putting $\psi = \pi/2$, $\theta = 0$ and $\psi = \pi/2$„ we obtain the rotation matrix of the world reference **N** with respect to camera

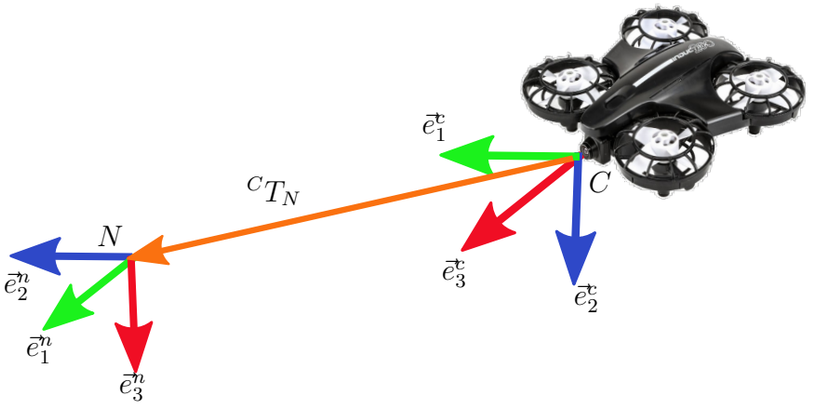


Figure 3.7 – Orthogonal vector representation.

reference **C**,

$$^{\mathbf{C}}R_{\mathbf{N}} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix} \tag{3.38}$$

This rotation can be also expressed in terms of quaternions using equation (**??**), which is a simple form to represent a rotation.

3.6.3 Camera-body-world transformation

The transformation between coordinates systems according to Fig. 3.6 is made using 4×4 matrices, then

$$^{\mathbf{C}}M_{\mathbf{N}} = \begin{bmatrix} ^{\mathbf{C}}R_{\mathbf{N}} & ^{\mathbf{C}}T_{\mathbf{N}} \\ 0_{1\times3} & 1 \end{bmatrix} \tag{3.39}$$

$$^{\mathbf{C}}M_{\mathbf{B}} = \begin{bmatrix} ^{\mathbf{C}}R_{\mathbf{B}} & ^{\mathbf{C}}T_{\mathbf{B}} \\ 0_{1\times3} & 1 \end{bmatrix} \tag{3.40}$$

taking into account that $^{\mathbf{N}}M_{\mathbf{C}} = (^{C}M_{\mathbf{N}})^T$, the complete transformation is given by:

$$^{\mathbf{N}}M_{\mathbf{B}} = ^{\mathbf{N}}M_{\mathbf{C}}\,^{\mathbf{C}}M_{\mathbf{B}} \tag{3.41}$$

The inverse transformation is given by (3.42), and it represents the transformation

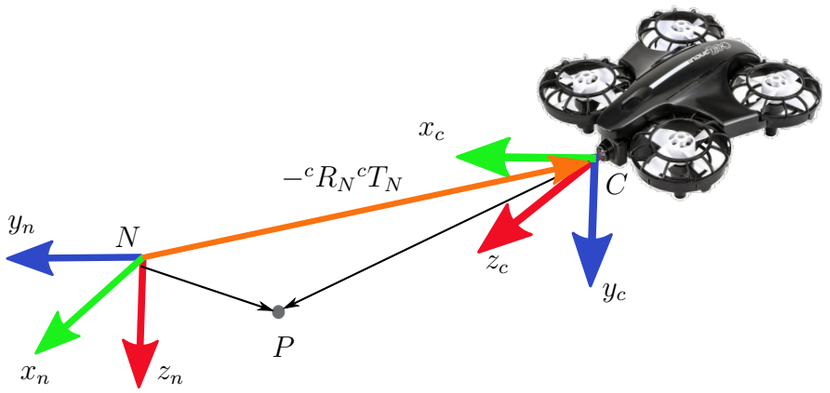


Figure 3.8 – Inverse transformation from the camera with respect to the world.

from the camera **C** with respect to the world **N**, see Fig. 3.8

$$ {}^{\mathbf{N}}M_{\mathbf{C}} = \begin{bmatrix} {}^{\mathbf{C}}R^T_{\mathbf{N}} & -{}^{\mathbf{C}}R^T_{\mathbf{N}}{}^{\mathbf{C}}T_{\mathbf{N}} \\ 0^T & 1 \end{bmatrix} \tag{3.42} $$

Considering the translation between axes is equal to zero the rotation of the body reference frame with respect to the world reference frame, taking into account a point acquired by a camera, is given by the rotational part of ${}^{\mathbf{N}}M_{\mathbf{B}}$.

3.7 Summary

This chapter was focused on the principal characteristics and operation of our quadrotor vehicle. The representation of position and attitude were also introduced to analyse how 3D pose describes the rigid body motion. Then, mathematical model of quadrotor system was presented. Finally, the relation between world, camera, and body frames to merge in to a complete system was presented.

Chapter 4
Camera pose estimation

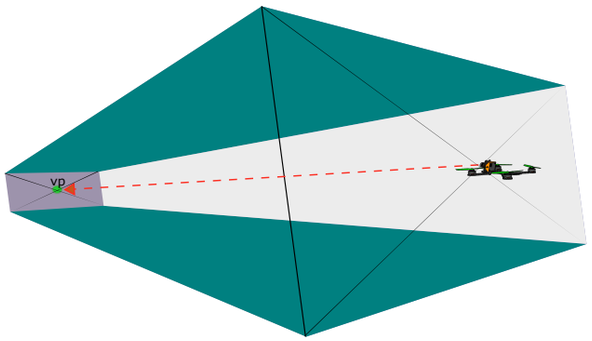
This chapter describes the proposed algorithm whose aim is to estimate certain position of an embedded camera in a quadrotor system. A description of the proposed configuration is analysed in Section 4.1. In section 4.2 position algorithm and its numerical validation are described. The chapter is concluded with a description of some problems encountered during the design, section 4.3.

4.1 Description system

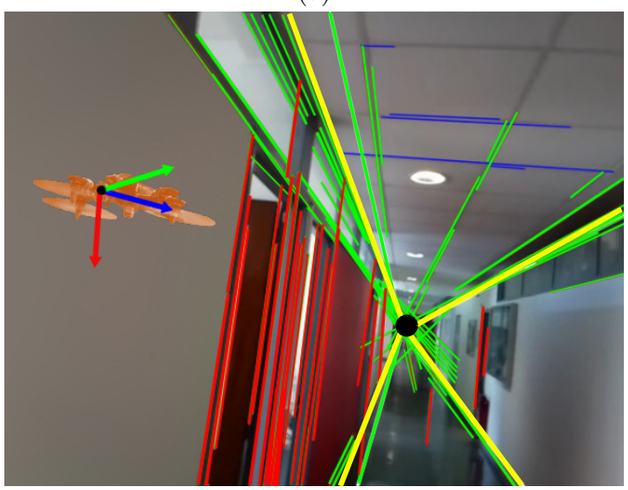
The proposed configuration takes advantage of the video acquired from a corridor. The image video is acquired by a camera over a quadrotor system. These images are taken while the vehicle is flying through a corridor, and considering that camera is always pointed to the end of the corridor, see Fig. 4.1a. As can be seen in Elloumi et al. [2014], and Boulanger et al. [2006a], our work also rely on the Manhattan-world assumption Coughlan and Yuille [1999], where different lines describe the floor, ceiling, walls, windows, and doors, see Fig. 4.1b. Moreover, several parallels lines, which describes the depth of a corridor, can be clearly seen under this perspective. These lines intersect in an infinity point named vanishing point vp. This particular vp has a direction which can coincide with orthogonal direction (z_c-axis) with respect to the projection center of the camera. Furthermore, according to its perspective, a common structured environment can be represented by their orthogonal directions which are extracted by the environment geometry. These directions are used to estimate each vector of the rotation matrix, see section 2.1.4. Since camera is supposed to be on-board of the quadrotor system, and the horizontal is kept during hover, see Fig. 4.1b, we can say that horizontal and vertical orientation of the lines yield the orthogonal direction of missing vectors. Once this data is estimated, and performing suitable frames transformation, head direction of the system can be controlled using a control law, which will be described in Chapter 5, by exploitation of the angle

yaw (ψ).

Besides, image perspective often affects the extracted lines in the real world having different angles depending on the direction of view camera view and its position in 3D space. Particularly some lines, which shapes comes from edges of a corridor, can give useful information to extract certain position of the system. Fig. 4.2c depicts the camera which is near to the center of the corridor, where a symmetry is observed on the opposite edge lines. On the contrary, when the camera is in a different place, this symmetry is lost due the variations in lines angles, see Fig. 4.2. Thus, we can deduce that if the collinearity on opposite edge lines is applied, the obtained data can be used like certain position which can be used to pose the quadrotor system in the world plane parallel to the image plane.



(a)



(b)

Figure 4.1 – Quadrotor flying into a corridor, the head-direction is given by a vanishing point, and the pose of the quadrotor is obtained geometrically using principal lines which are estimated using data from the corner borders.

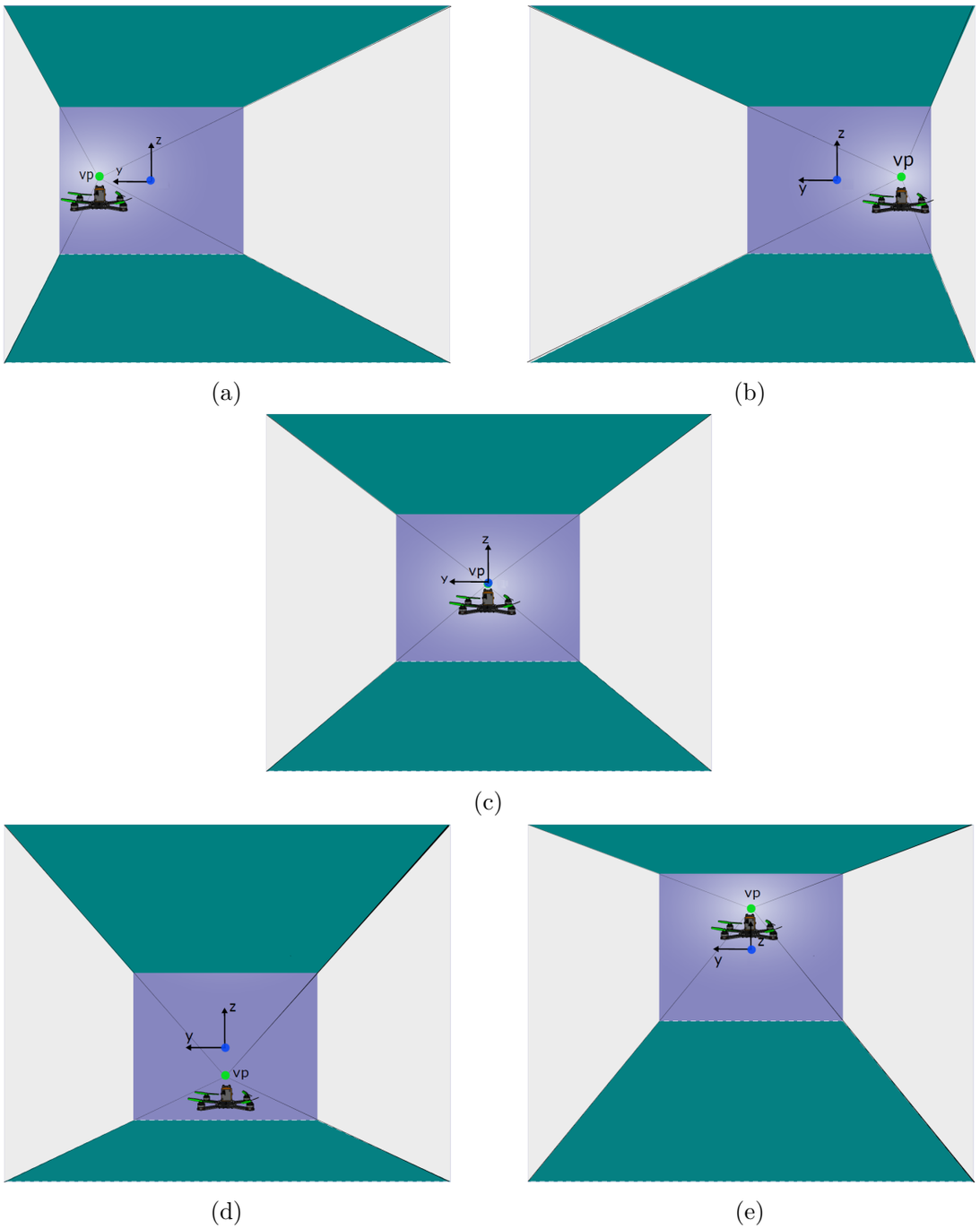


Figure 4.2 – The origin of the world is supposed to be in the corridor center, and the vanishing point *vp* can be moved in agreement of the camera projective perspective. There is a variation of the angles of each corner line, which is used to compute the position error. The quadrotor is placed in the corridor: c) center, a) near to the left wall, b) near to the right wall, d)down, and e) up.

4.2 Pose estimation algorithm

A general scheme of our proposed algorithm can be seen in Fig. 4.3. The strategy shows a sequence of steps to estimate certain position according to the environment. Firstly the image is acquired using a single camera. After, the line detector EDLines is applied to obtain the lines which describes the environment. Then, the detected lines are grouped in accordance to the slope resulting into three subsets of lines (vertical, horizontal, and diagonal). Next, from one finite vanishings point and two infinite vanishing points can be extracted using the classified lines. Thereafter, camera rotation matrix is estimated performing an algorithm to extract the camera parameters. Later, a general least squares fitting is applied to extract four edges lines. This task is achieved using a set of points describing the edges of a corridor. Finally, the collinearity in opposite edges lines is applied to estimate the camera pose on the plane image. This collinearity yields certain values according to camera position in the 3D space of a corridor. Fig. 4.4 shows the general procedure, where the proposed algorithm is applied to an image taken by embedded camera. This figure shows all the lines acquired by detector lines. The extraction of the four edge lines is also represented. This algorithm was inspired from ideas of Akinlar and Topal [2011], Boulanger et al. [2006b] and Lee et al. [2009]

4.2.1 Dominant lines detection

While edges and general curves are suitable for describing the contours of natural objects, straight lines describes a typical man-made world which is full of it. Thus, detecting and matching lines can be useful in a variety of applications, including architectural modeling, pose estimation in urban environments, and so on. In the case of a common corridor, lines can be classified as vertical lines L_v, horizontal lines L_h. We name the rest of lines as diagonal lines L_d. Since the image from camera gives the projection of a real world, the direction of lines describing a corridor can be matched with the orthogonal direction of the world system.

From implementation point of view, the computational time is considered a constraint because this time has to be taken into account in the sample rate. Therefore, the performance of the algorithm has to be designed in such a way that it has low computational cost. Normally, the algorithm of detection lines is the most expensive in terms of computational cost; for instance the Hough transformation , see Hough [1959], Duda and Hart [1972b], is the most common feature extraction technique. However, it has some drawbacks: Its parameters are sometimes quite difficult to tuning, and also a detection edge (Canny, Sobel, etc) is required before to run the algorithm. Otherwise, in literature there are some alternatives, for instance, the line segment detector called EDlines Akin-

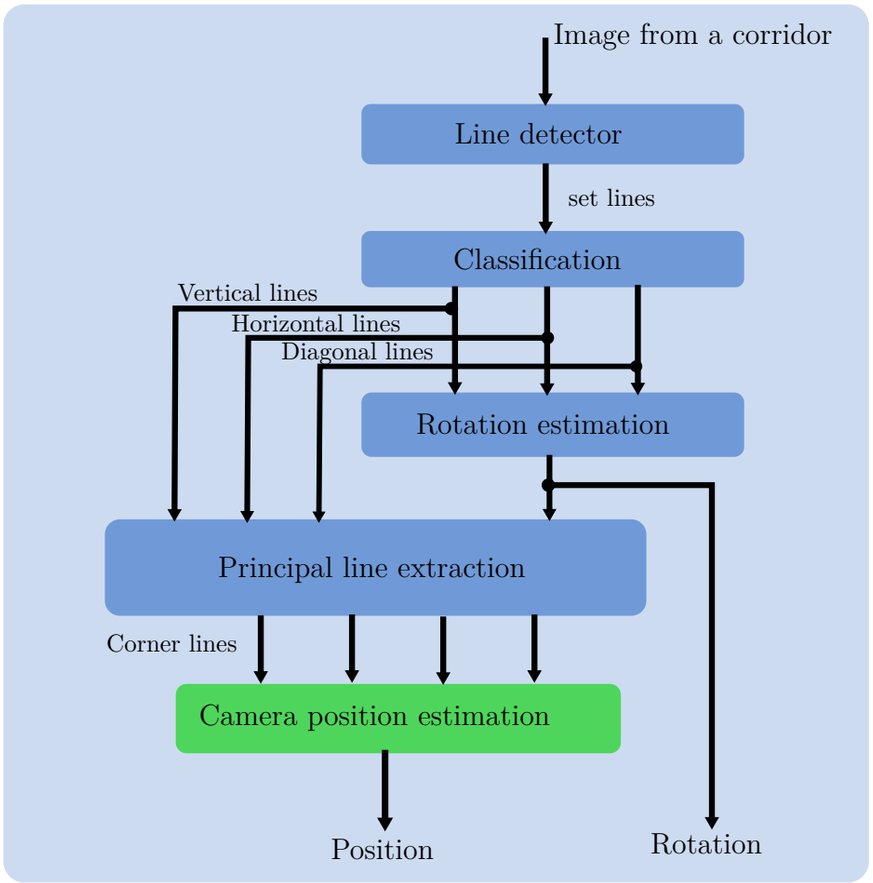


Figure 4.3 – Pipeline of position camera estimation.

lar and Topal [2011] which gives accurate results, and also has low time process (all the segments for an image 240×320 are computed in 3 ms). This line detector makes use of the clean, contiguous (connected) chain of edge pixels produced by an edge detector, the Edge Drawing (ED) algorithm Topal and Akinlar [2012]; hence the name EDLines. The detector includes a line validation step which lets it control the number of false detection. With its accurate results and low computational cost, this algorithm is very suitable in a real time implementation. This algorithm can be seen as a function, which has as input the image, and as output an array with all the lines, see Fig. 4.5. The acquired information is a unclassified set of initial and final points of each line.

4.2.2 Lines classification and vanishing point extraction

In Chapter 2, we had seen that the perspective projection changes the tilt of lines which are describing the environment according to position of the camera. Thus, assuming that the camera takes an image in direction toward to the end of a corridor (as can be see in

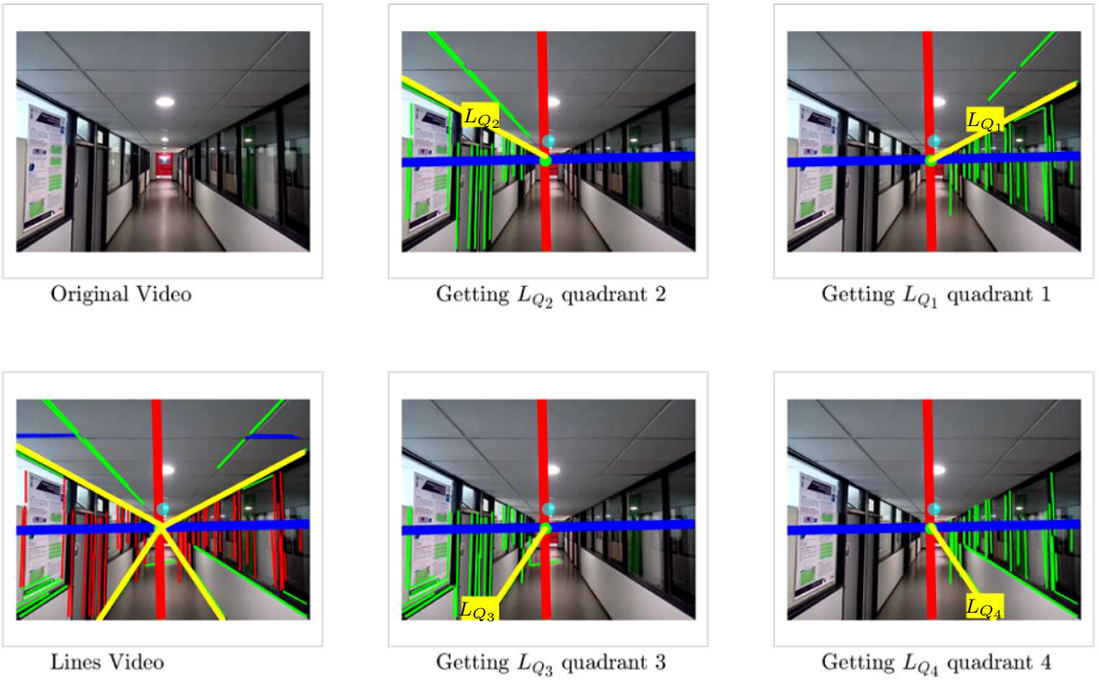


Figure 4.4 – These images introduce the applied methodology. The scene is a corridor and an image is taken from the video to extract a set of lines that could describe geometrically the environment and the rotation matrix. Four new subsets of lines are made taking into account the vanishing point (quadrant 1, 2, 3 and 4). The information of each quadrant is used to obtain the principal lines (L_{Q_1}, L_{Q_2}, L_{Q_3}, L_{Q_4}).

Fig. 4.5), this view is matched with the perspective projection where one finite vanishing point is enable, see section 2.1.4. The lines detected by EDLines are a set of lines which can be grouped in three sets that are named horizontal lines L_h, vertical lines L_v and diagonal lines L_d. Each set of lines is used to obtain the vanishing points. Therefore, diagonal lines are used to estimate the finite vanishing point vp, and the direction lines of L_h and L_v are used to extract the infinite vanishing point. The vps direction, which normally are matching with the frame reference of real world, are known as orthogonal vps direction, which can be employed to rebuild the camera rotation matrix.

Lines classification

Lines classification is based in making a classification using the inclination of each line. Once the slope of each line $\mathbf{L}$ is available, a new lines subsets (L_h, L_v, and L_d) can be performed. Lines direction are given from -90 to 90 degrees, and taking in account that camera keeps the horizontal position, the sub-classification can be extracted as follows:

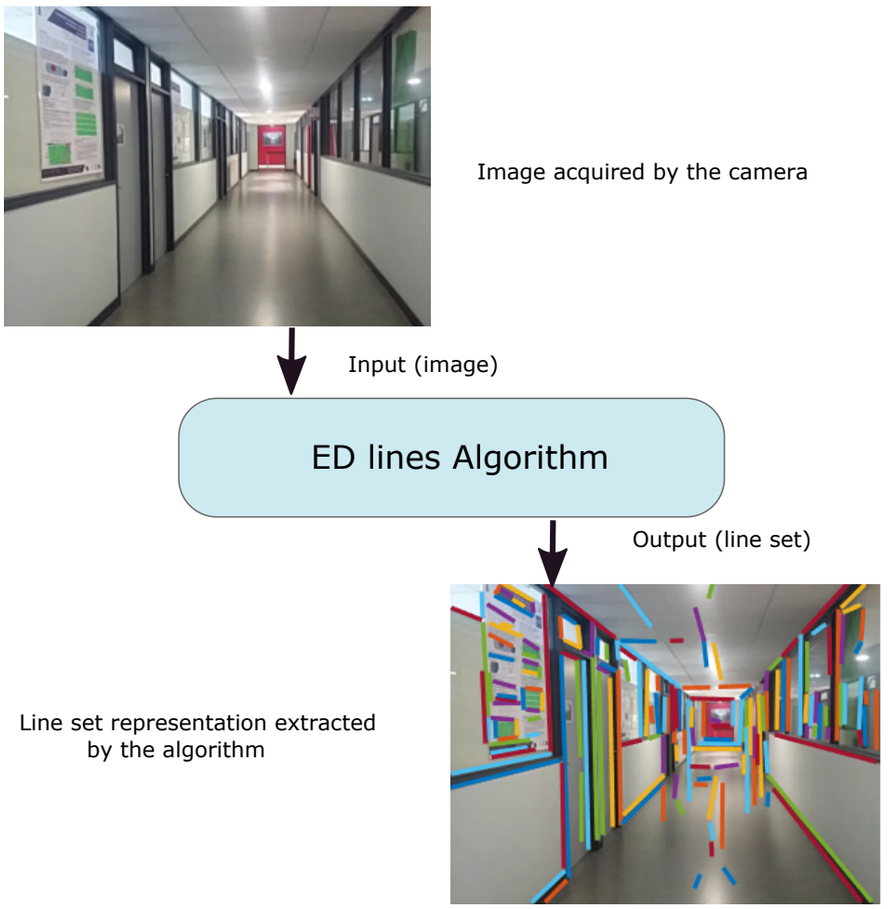


Figure 4.5 – The EdLines algorithm is taken as a function where the input is the image acquired by the camera, and the output is a lines set. Each line has as information the initial and final point.

$$\mathbf{L} \quad = \quad \begin{cases} 110 \geq \mathbf{L} \geq 80 & L_v = \mathbf{L} \\ -10 \geq \mathbf{L} \leq 10 & L_h = \mathbf{L} \\ \text{otherwise} & L_d = \mathbf{L} \end{cases} \tag{4.1}$$

After the sub-classification, the lines are graphically represented with a color according the new sub-classification. Fig. 4.6 depicts it, where the blue, red and green lines represent horizontal, vertical and diagonal lines respectively.

Vanishing point: Finite and infinite vanishing point

In the frame of indoor navigation, the main orthogonal directions in a corridor environment consist generally in a vertical direction (often associated with one infinite vp) and two horizontal directions. All detected lines from a corridor can be employed to estimate

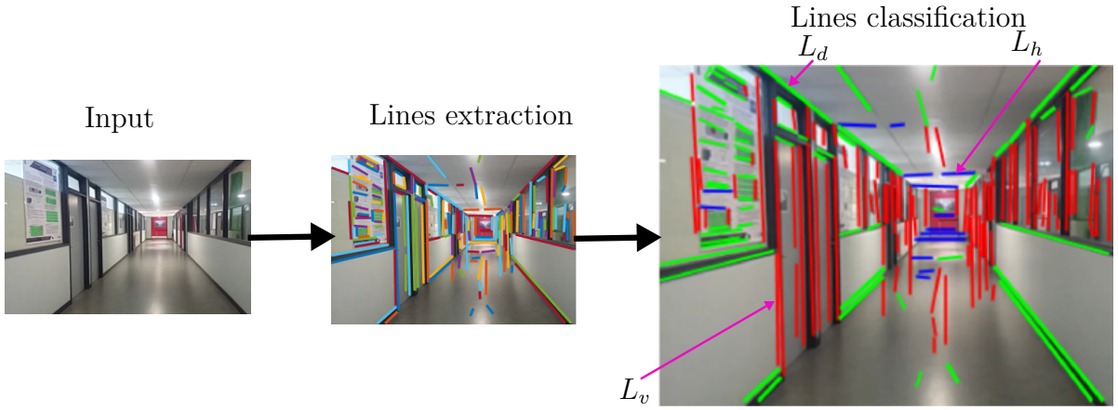


Figure 4.6 – Lines classification according to the slope of each line. Vertical lines are represented in red, horizontal lines are in blue, and diagonal lines are in green.

the *vp*s. However, the given information contains plenty of outliers that could contaminate the final estimation. In order to find such an orthogonality in the right *vp*s, RANSAC [1] is used to prun the undesired data.

The goal is to spot orthogonality between these set-lines. In this particular case, the camera is supposed to be parallel to the horizontal, and its direction is pointed through the end of the corridor. Therefore, one finite and two infinite *vp*s can be established using this configuration. The *vp* extraction is carried out in two steps: first, the finite *vp*s are detected; and then the infinite *vp*s are identified.

Finite vanishing point

In section 2.1.5 was shown this particular case. The algorithm begins selecting randomly two lines (L_{d_i}, L_{d_j}) from the set L_d for generating a vanishing point hypothesis vp_h. The participants lines fitting the best vanishing point inside a neighborhood are computed using a consensus score given by equations

$$score = \sum_{i=0}^{n} \Upsilon(vp_h, l_i) \tag{4.2}$$

where n is the number of dominant lines of the subset $(L_d, L_h, L_v,)$ and

$$\Upsilon(vp_h, l_i) = \begin{cases} 1 & d(vp_h, l_i) < \delta \\ 0 & \text{otherwise} \end{cases} \tag{4.3}$$

1. RANSAC (Random Sample Consensus) has become a simple and powerful method to provide a partition of parallel straight lines into clusters by pruning outliers.

with $d(vp_h, l_i)$ is the Euclidean distance from the finite vp candidate vp_h to the line l_i $(L_d, L_h, L_v,)$. δ is a fixed threshold given in pixels.

Previous steps need to be repeated k times which is the number of random combinations for selecting a pair of line segments from L_d. The consensus score is based on a distance between the candidate straight line and the intersecting point, eq. (4.3). All lines whose distance is below a fixed threshold δ are considered as participants. These lines are used to estimate the finite vanishing point which according to environment (corridor) is normally near to the center of the image plane.

Infinite vanishing point

The procedure used to detect two infinite vps is similar to that of finite vp except that the aim is to identify the vanishing line direction instead of its intersection. For this, a line segment is selected randomly from the remaining dominant lines of the set L_v or L_h. Then, its direction is computed using the consensus score given by eq. (4.2) and eq. (4.4).

$$\Upsilon(v, l_i) = \begin{cases} 1 & Min(\widehat{(\overrightarrow{v}, \overrightarrow{l_i})}, \widehat{(\overrightarrow{l_i}, \overrightarrow{v})}) < \iota \\ 0 & \text{otherwise} \end{cases} \tag{4.4}$$

where $\widehat{(\overrightarrow{v}, \overrightarrow{l_i})}$ is the angle between the infinite vp direction from the image center, and the line l_i is used to test in image space. In this case, the consensus uses an angular distance between the direction of the candidate straight line and the direction representing the infinite vp eq. (4.4). The threshold ι is given in degrees.

At the end of the process one finite vanishing point and two infinite vanishing points are identified, which are the vertical vp and the two horizontal vps. Fig. 4.7 represents the vanishing point which is the intersection of the set of L_d, and the vertical and horizontal lines direction of infinite vanishing points are represented with the lines in red and blue respectively.

4.2.3 Rotation matrix estimation using vanishing points

Even though intrinsic and extrinsic camera parameters can be extracted using an algorithm based on vanishing points, in this work the intrinsic parameters are considered to be known. Then, the task is focused on computing extrinsic parameters.

Camera rotation matrix, which describes the orientation of the camera with respect to the new world coordinate system, transforms points from real world to the image plane. Its columns are vectors of the world reference frame $\mathbf{N}[\vec{e}_1^n, \vec{e}_2^n, \vec{e}_3^n]$ expressed in the camera

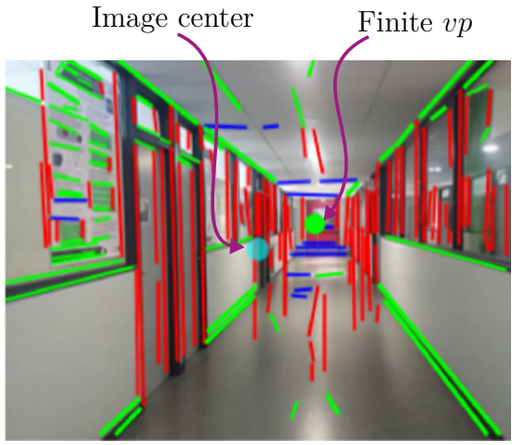


Figure 4.7 – Representation of *vp* and main point; Vanishing point is acquired using RANSAC; image center is given by the intrinsic parameters.

reference frame **C**. In other words, world reference frame is rotated to make it parallel to the camera reference frame, as it was presented in section 3.6. The three directions of vanishing points, having as origin the optical center of the camera, are assumed to be orthogonal. Thus without loss of generality, the following relations have to be satisfied for the final calibration.

$$f > 0 \tag{4.5a}$$

$$\vec{e}_1^n \cdot \vec{e}_2^n = \vec{e}_2^n \cdot \vec{e}_3^n = \vec{e}_3^n \cdot \vec{e}_1^n = 0 \tag{4.5b}$$

$$\|\vec{e}_1^n\| = \|\vec{e}_2^n\| = \|\vec{e}_3^n\| \tag{4.5c}$$

where f is the focal length. Fig. 4.8 depicts an ideal case where horizontal, vertical, diagonal lines and the finite *vp* can be extracted in a corridor.

The obtained lines from the image corridor provide enough information to estimate the extrinsic parameters where rotation of world reference frame **N** w.r.t. camera reference frame **C** can be extracted, see section 3.6. That is, as the camera is supposed to be in an horizontal position and pointed to the end of the corridor, orthogonal directions of the environment can be extracted. This rotation is given as the rotation to world (perspective of the real world-image plane) with respect to the camera ${}^{\mathbf{C}}R_{\mathbf{N}}$.

A finite vanishing point, two infinite vanishing points

Normally, the corridor environment is defined by one finite vanishing point and two infinite vanishing points. This case occurs when two axes of the world frame are parallel to the image plane. Fig. 4.8 introduces the terms involved for the camera calibration using

one finite vanishing point $\overrightarrow{0vp} = [vp_x, vp_y, -f]^T$ and two infinite vanishing points with directions $\vec{I_1} = [I_{1_x}, I_{1_y}, 0]^T$ and $\vec{I_2} = [I_{2_x}, I_{2_y}, 0]^T$. 0 represents the origin of camera reference frame **C**.

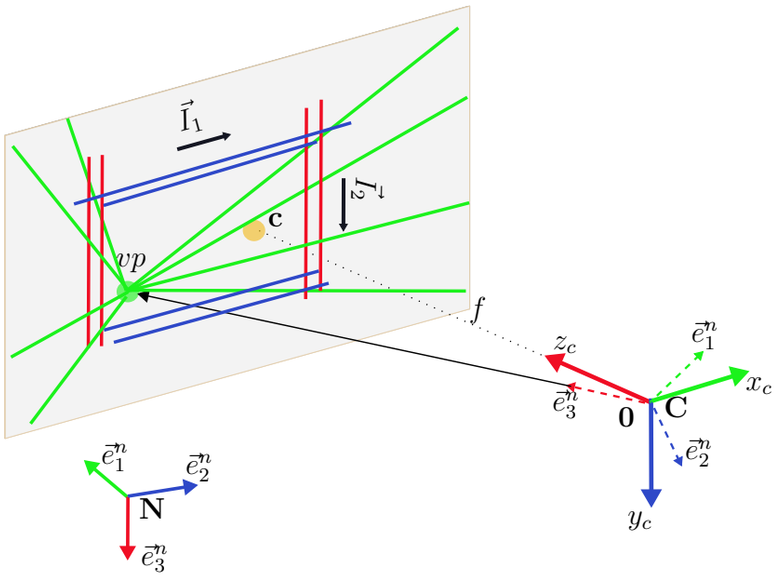


Figure 4.8 – Camera calibration with one finite and two infinite vanishing points. $[x_c, y_c, z_c]$ is the camera coordinate frame. vp is the finite vanishing point. $\vec{I_1}$ and $\vec{I_2}$ represent the infinite vanishing point directions. **c** depicts the main point of the image, and $[\vec{e}_1^n, \vec{e}_2^n, \vec{e}_3^n]$ represents the rotation of the world with respect to camera.

The vector e_3^n defines a non-normalized form of $\vec{e}_3^n$, and is computed from the finite vanishing point as

$$e_3^n = [e_{3_x}^n, e_{3_y}^n, e_{3_z}^n]^T = \overrightarrow{0vp} = [vp_x, vp_y, -f]^T \tag{4.6}$$

Horizontal direction

The coordinate axis $\vec{e}_1^n$, as indicated in Fig. 4.9, lies on the plane defined by the vector $\overrightarrow{0c}$ and the direction $\vec{I_1} = [I_{1_x}, I_{1_y}, 0]^T$. Then e_1^n, the non-normalized version of $\vec{e}_1^n$, can be expressed as $e_1^n = [I_{1_x}, I_{1_y}, e_{1_z}^n]^T$. The goal is that $\vec{e}_1^n$ and $\vec{e}_3^n$ belong to an orthogonal coordinate frame, hence

$$e_1^n \cdot e_3^n = I_{1_x} e_{3_x}^n + I_{1_y} e_{3_y}^n + e_{1_z}^n e_{3_z}^n = 0 \tag{4.7}$$

therefore

$$e_{1_z}^n = \frac{I_{1_x} e_{3_x}^n + I_{1_y} e_{3_y}^n}{f} \tag{4.8}$$

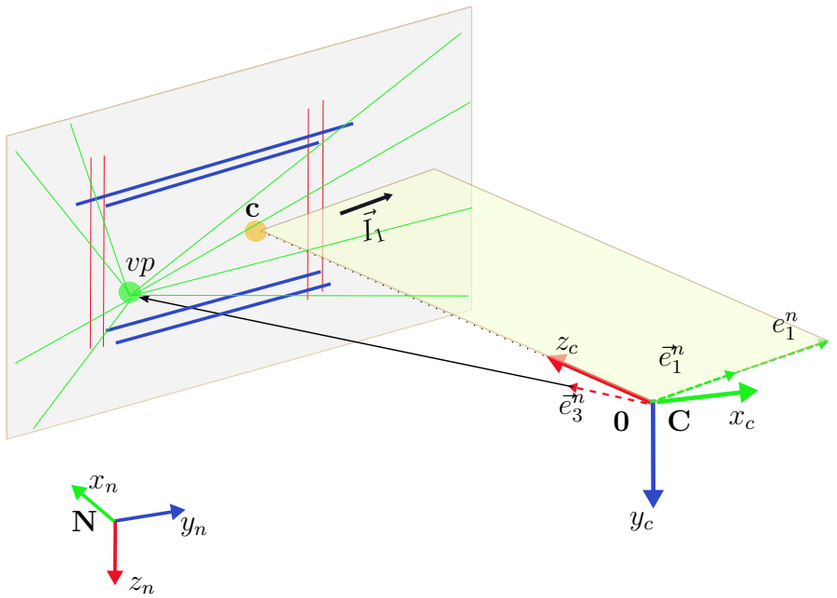


Figure 4.9 – $\vec{e}_1^n$ estimation from the infinite vanishing points.

Vertical direction

Similar procedures can be used for $e_2^n = [I_{2_x}, I_{2_y}, e_{2_z}^n]^T$ (see Fig. 4.10), thus

$$e_{2_z}^n = \frac{I_{2_x} e_{3_x}^n + I_{2_y} e_{3_y}^n}{f} \tag{4.9}$$

Typically the vertical lines are plenty and more precise in many images, thus it retains e_2^n and e_3^n and compute e_1^n using a cross product, $e_1^n = e_2^n \times e_3^n$. Thus rotation of world reference frame **N** w.r.t. to camera reference frame **N** is given by

$${}^{\mathbf{C}}R_{\mathbf{N}} = \left[\vec{e}_1^n, \quad \vec{e}_2^n, \quad \vec{e}_3^n\right] \tag{4.10}$$

This rotation matrix, which meets the conditions from eq. (4.5a), is obtained by normalizing e_1^n, e_2^n and e_3^n.

4.2.4 Principal line extraction

In the beginning of the chapter it was mentioned that the perspective of a corridor has some changes depending on the camera position. These perspectives were shown in Fig. 4.2 according to the camera position in 3D space of a virtual corridor. Images present a phenomenon on corners lines due to the perspective. Fig. 4.4 depicts this situation using a real image of a corridor. In the images it can be seen that corners lines have a variation

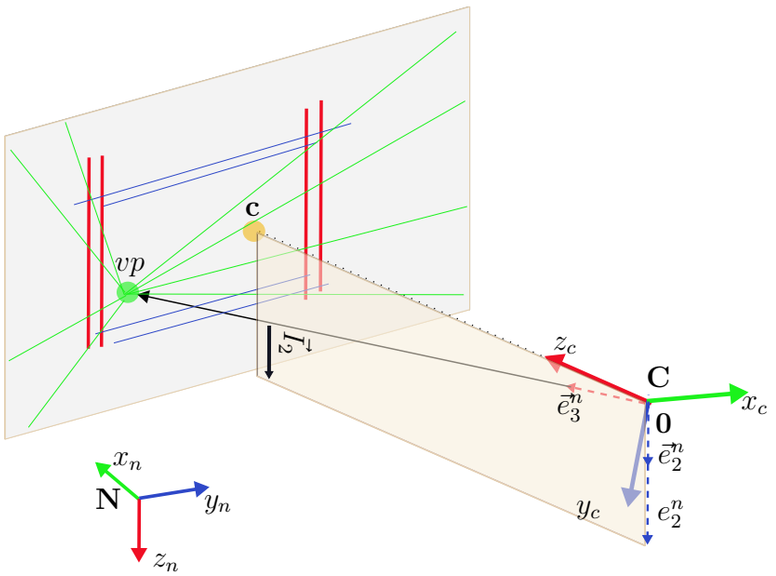


Figure 4.10 – $\vec{e}_2^n$ estimation from the infinite vanishing points.

on their angles according to the camera position in 3D space of the corridor. This variation can be used to obtain some information to know the position under two restrictions:

- Camera head direction has to be pointed to the end of the corridor,
- The system has to be always posed close to the horizontal.

Then, four lines, representing the corners along of a corridor, can be extracted to obtain a particular information which can be used to pose the system in the center of a corridor. In Fig. 4.11, the camera rotation is visually represented in the image by the thick lines which are intersected on the *vp*. This new virtual coordinate frame represents the camera reference frame with respect to the world reference frame whose origin is in the center of the corridor. This virtual frame follows the movements of the camera, and its movements are considered to extract the edges lines. The process followed is as follows:

- Firstly, the image is partitioned in four quadrants bounded by the coordinates of *vp* and the variation of rotation matrix,
- Then, as the vertical L_v, horizontal L_h and diagonal lines L_d are concentrated in the corner of the corridor, a neighborhood of points, describing each edge line, can be used to estimate the corner line of each quadrant, see Fig. 4.11
- Finally, edge (corners) lines are estimated using General Least Squares fitting.

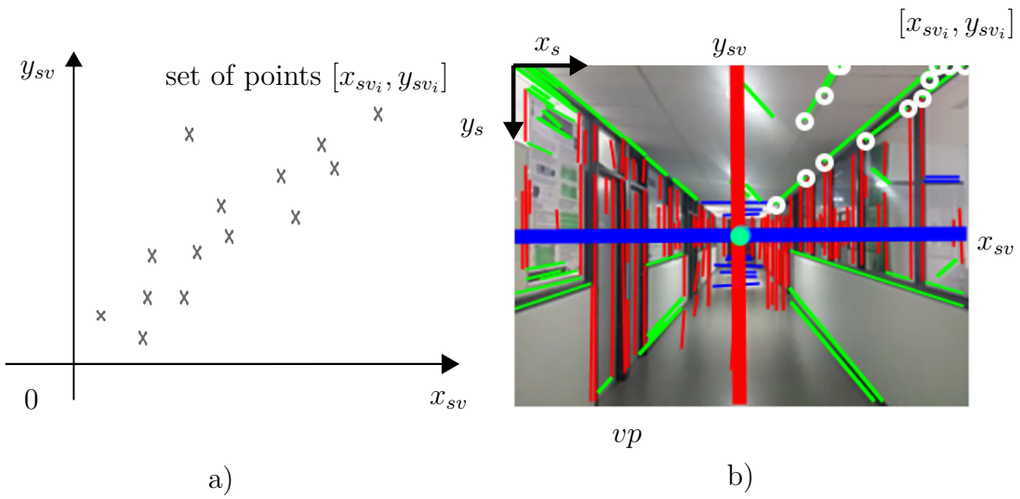


Figure 4.11 – a) Set of points to estimate a line in a coordinate system, b) set of point (x_{sv_i}, y_{sv_i}) of the frame image plane, vanishing point vp, and vertical line y_{sv} and horizontal line x_{sv}.

Since all the lines are clustered in four quadrants and centered with respect to the vp, these data can be used like a dynamic coordinate system whose axes x_{sv} and y_{sv} take direction of the infinite vertical line and the infinite horizontal line, see Fig. 4.11.

General least squared GLS: principal line estimation

The neighborhood of points describing the edges of the corridor has been used to extract four lines. These lines have to be estimated during each frame in order to obtain a dynamic position. That means, the algorithm is turned according to the frame rate. To accomplish this task, the general least squared algorithm has been applied. The process to make it is as following:

- Firstly, given the information proportioned by EDlines algorithm, points from diagonal, vertical, and horizontal lines are taken as main work data. These points are given w.r.t. the coordinates of the image plane $[x_s, y_s]$.

- Then, a transformation is made to center these data taking as origin the finite vanishing point vp.

$$x_{sv_i} = x_{s_i} - vp_{x_m} \tag{4.11}$$

$$y_{sv_i} = y_{s_i} - vp_{y_m} \tag{4.12}$$

where $[x_{s_i}, y_{s_i}]$ represent coordinates of initial and final point from each set of lines,

and i describes the numbers of lines taken from each frame. $[vp_{x_m}, vp_{y_m}]$ are the coordinates of the vanishing point. The frame number is represented by m.

- Now having a set of points $[x_{sv_i}, y_{sv_i}]$, which are represented as circles in Fig. 4.11, the GLS is applied to obtain the principal line. Thus, the covariance matrix is given by

$$C_m = \begin{bmatrix} \sum_i \hat{x}_i^2 & \sum_i \hat{x}_i \hat{y}_i \\ \sum_i \hat{x}_i \hat{y}_i & \sum_i \hat{y}_i^2 \end{bmatrix} \tag{4.13}$$

where $\hat{x}_i$ and $\hat{y}_i$ are given by

$$\hat{x}_i = x_{sv_i} - \bar{x} \tag{4.14}$$

$$\hat{y}_i = y_{sv_i} - \bar{y} \tag{4.15}$$

and $\bar{x}$ and $\bar{y}$ represent the arithmetic mean, of x_{sv_i} and y_{sv_i} respectively.

- Finally the principal line is estimated computing the eigenvalues (a_m, b_m) from the covariance matrix C_m, and the principal line of each quadrant L_{Q_k} is given by

$$y_{L_{Q_k}} = -\frac{b_m}{a_m} x_{L_{Q_k}} \tag{4.16}$$

where $x_{L_{Q_k}}$ takes the values of x_{sv} depending of each quadrant $(k = 1, 2, 3, 4)$. Fig. 4.12 shows the principal line estimated from the neigboorhood of points of the quadrant 1.

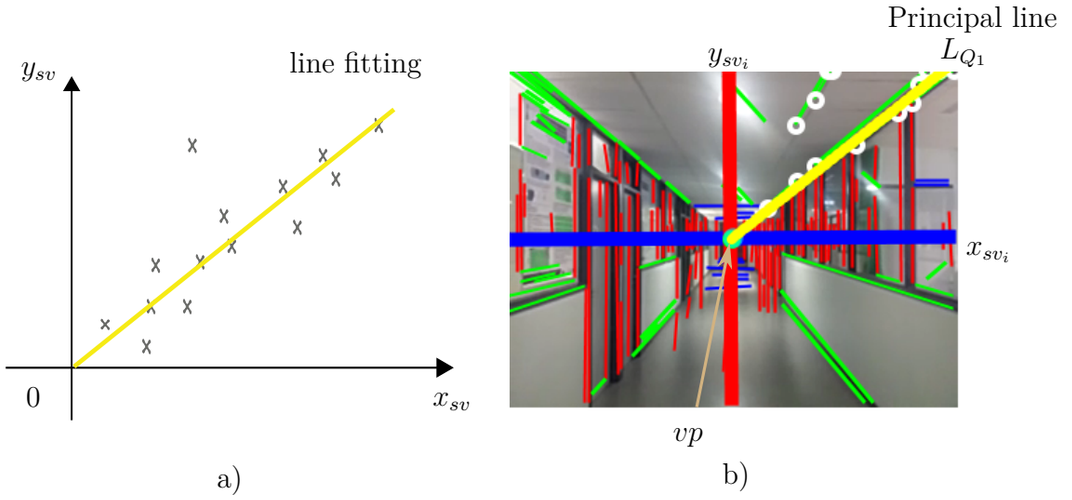


Figure 4.12 – a) Fitting line using total least squares, b) Principal line L_{Q_1} using data of the image plane.

Weighted total least squares

The above least squares formulation assumes that all feature points are matched with the same accuracy. However, the images contain plenty of undesired data resulting in a noisy estimation. Thus, if a scalar variance estimate σ_i^2 is associated with each correspondence, a weighted least squares problem can be minimized instead,

$$E_{WLS} = \sum_i \|\mathbf{r}_i\|^2 \tag{4.17}$$

The covariance matrix can be rewritten as

$$C_m \equiv \begin{bmatrix} \omega_m \sum_i \hat{x}_i^2 & \omega_m \sum_i \hat{x}_i \hat{y}_i \\ \omega_m \sum_i \hat{x}_i \hat{y}_i & \omega_i \sum_i \hat{y}_i^2 \end{bmatrix}$$

This matrix is used for computing the eigenvalues and later the normal of each line. The previous strategy is done for each frame; however, to initialize the algorithm a line with 45 deg of inclination in each quadrant is considered for implementation. For estimating the new line, the previous line is taken into account, with this, ω_m can be defined,

$$\omega_m = e^{\frac{-0.5\mu^2}{\sigma^2}}$$

where σ defines the variance parameter and,

$$\mu = \arctan \frac{x_{sv_i} a_{m-1} + y_{sv_i} b_{m-1}}{x_{sv_i} a_{m-1} - y_{sv_i} b_{m-1}}$$

with a_{m-1}, b_{m-1} represents initial line hypothesis. ω_m also represents the weight that supplies the response variance to a constant value.

4.2.5 Position camera estimation using principal lines

Taking into account the definition of collinearity, which is known as a set of points having the property that they are lying on a single line, lines obtained in previous section can give us certain useful information to know the position of a camera. It should be noted that lines are extracted in a particular environment (corridor). Thus, this situation can be observed in Fig. 4.2 where lines describing a corridor are changing its slope according to the camera perspective or camera position in 3D space. This phenomenon happens when camera head direction is pointed toward the end of the corridor. In Fig. 4.4 edges lines ($L_{Q_1}, L_{Q_2}, L_{Q_3}, L_{Q_3}$) can be observed which are intersecting on the *vp*. Using opposite edges lines, that is (L_{Q_1}, L_{Q_3}) and (L_{Q_2}, L_{Q_4}), a collinearity can be computed between

these mentioned lines, and it can be expressed as

$$CP_{1,3} = L_{Q_1} \wedge L_{Q_3} \tag{4.18}$$
$$CP_{2,4} = L_{Q_2} \wedge L_{Q_4}, \tag{4.19}$$

$CP_{1,3}$ and $CP_{1,3}$ represent the collinearity between edges lines. $\wedge$ represents the cross product. L_{Q_1}, L_{Q_2}, L_{Q_3}, L_{Q_4} are the vectors describing each line assuming that its third element is equal to zero.

The result obtained by equations 4.18 and 4.19 are two vectors where the first and second element is approximately equal to zero. The third element is normally different to zero when collinearity is not present. This result gives certain data which can render us a relation to the camera movements with respect to the image plane of a corridor. To accomplish this, we use the function *sign* to extract the sign of the third element of $CP_{1,3}$ and $CP_{1,3}$. Then, we normalize the vectors $CP_{1,3}$ and $CP_{1,3}$. Finally the proposed relations are given by

$$y_r = sign(CP_{1,3}(3))||CP_{1,3}|| - sign(CP_{2,4}(3))||CP_{2,4}|| \tag{4.20}$$
$$z_r = sign(CP_{1,3}(3))||CP_{1,3}|| + sign(CP_{2,4}(3))||CP_{2,4}|| \tag{4.21}$$

where y_r and z_r are the data which can be seen as the positions with respect to center of the corridor. A numerical validation is presented in the next section. $sign(-)$ is sign function, and $||-||$ represents the norm.

4.2.6 Numerical validation

The previous algorithm for pose estimation was firstly tested off-line. For corroborating equations (4.18) and (4.19) a sequence of images of a real corridor were taken with specified movements. These motions are shown in Fig. 4.13 and Fig. 4.14.

1) 2) 3) 4) 5)

Figure 4.13 – Pictures with a camera in the middle of a corridor with up and down movements. In this sequence: Starting from left side 1) stay at the middle, 2) move up, 3) returns to the middle, 4) move down and 5) returns to the middle.

1) 2) 3) 4) 5)

Figure 4.14 – Pictures with a camera in the middle of a corridor with left and right movements. In this sequence: 1) stay at the middle, 2) move right, 3) returns to the middle, 4) move left and 5) returns to the middle.

The experiment was about puting the camera in the middle of the corridor and moves it firstly up and down (Fig. 4.13) and after right and left (Fig. 4.14). These movements can be represented via edges lines in the image and observed graphically in Fig. 4.15a and Fig. 4.16a. On one hand, notice for example from Fig. 4.15a that both cross products have the same behavior and if the camera is moved up the lines representing $L_{Q_1} \wedge L_{Q_3}$ and $L_{Q_4} \wedge L_{Q_2}$ increase and when the camera is moved down they decrease, thus demonstrating the desired behaviour. On the other hand, when the camera is moved right and left, then the cross product between $L_{Q_1} \wedge L_{Q_3}$ and $L_{Q_4} \wedge L_{Q_2}$ are different (see Fig. 4.16a), nevertheless this information gives necessary data to estimate the y_r position, as it can be seen in Fig. 4.16b. In this figure it is observed that when the camera is moved right the y_r estimation is negative and if it is moved left, then a y_r positive response is obtained. In addition, it can be noted that z_r estimation remains quasi-constant. Similarly for Fig. 4.15b where with the up and down camera's movements the z_r state is estimated, it is also observed that y_r estimation is also quasi-constant.

In other words, when the camera is positioned in the center of a corridor, and is also pointed to end of it, the collinearity between opposite edges lines is zero. If the camera is in a different position, the result of collinearity between opposite edges lines is different to zero. So, these values can be employed to obtain certain data which can be used as input error data to close the loop of position control.

Fig. 4.17 and Fig. 4.18 show several images taking from a corridor using a smartphone camera, and also depict the results after that the proposed vision algorithm was applied. All the images on the left side are the images extracted by a smartphone camera without any processing. Detected lines from the environment geometry and the finite vanishing point are depicted in the middle images. And finally edge lines are shown in each image on the left.

The algorithm was applied in several corridors to see the behavior of the proposed method. For instance, it can be seen that Fig. 4.17b and Fig. 4.18c show a corridor smaller than the rest of figures. The presence of columns or rooms along the corridor are

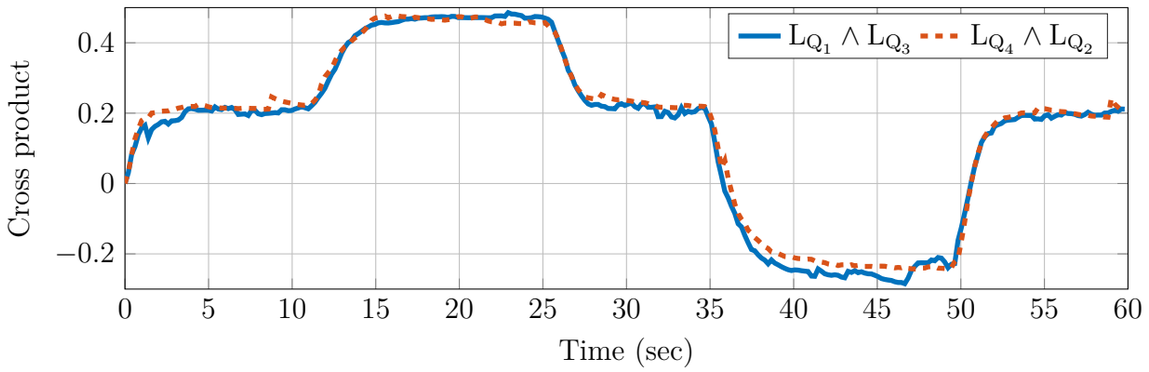
depicted in Fig. 4.17a and Fig. 4.17c. Moreover, a low illumination through the corridor is shown in Fig. 4.17d and Fig. 4.18a. Furthermore Fig. 4.17e, Fig. 4.18b, and Fig. 4.18d are obtained from a corridor with a large size having a natural illumination. In spite of these different scenarios, the behavior of the algorithm has been effective to obtain edges lines from different corridors.

Rotation

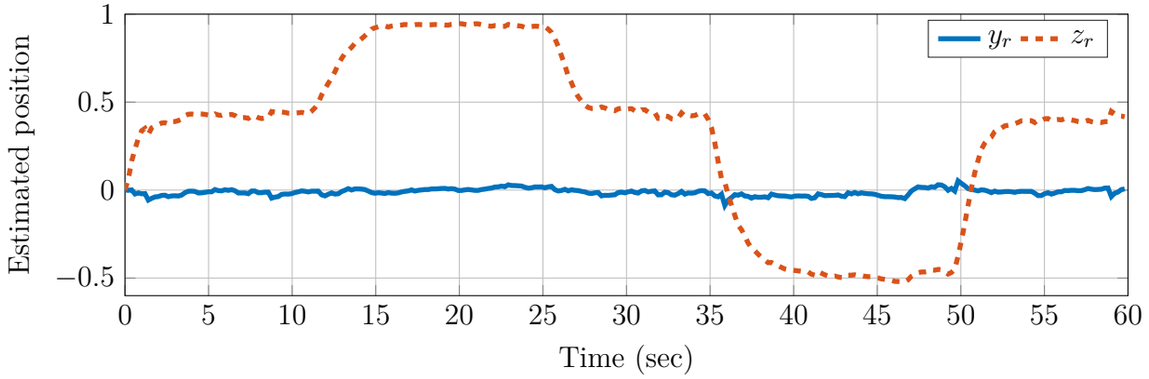
Even though the rotation matrix is estimated by the vision algorithm, this information is only used to feedback the attitude control in the head-direction. And once the direction is always controlled, position algorithm can be accomplished. Thus, a test was realized to probe our proposed algorithm. However, some constraints have to be into account in order to achieve the experiment. Then, since the data obtained into the corridor by a camera can not be compared directly by real data (particularly a motion capture system), an scenario was proposed at firsts in order to compare the real data with the estimated data (rotation matrix).

The scenario was the following: Firstly, the idea was to project an image on the wall. This image was taken manually using a camera from an smartphone in the middle of a corridor. Then, once the fixed image was projected on the screen of the camera, whose movement was produced by hand, was used to acquire the image projected on the screen. The data obtained by the camera was used to simulate the effect produced by the perspective as if the camera was capturing a real image from a corridor. At the same time, the camera movements were monitored by the Vicon system. The rotation matrix was estimated using the equations given in section 4.2.3. Fig. 4.19 depicts the sequence images from this test.

Fig. 4.20 shows the experimental test. Each data is labeled as moca data or camera data. The movement in yaw angle ψ can be observed in Fig. 4.20a, where it can be seen a variation of -20 to 20 degrees at time 16s and 23s. The ψ_{camera} signal follows accurately the ψ_{Moca} signal. Fig. 4.20b depicts the variation of roll angle. In time 23s, there is a movement induced by the movement in yaw angle, and in time 40s to 55s there is a direct motion in roll angle. Finally, the pitch data for both systems are represented in Fig. 4.20c. At time 35s-38s is observed that θ_{camera} not follows the signal of θ_{Moca}. This problem was a result of a lost of the projected image. That is vision algorithm was not capable to estimate the variation in this angle. However the last value of the angle was kept until a



(a) Frames position and cross product, the camera is moving up and down



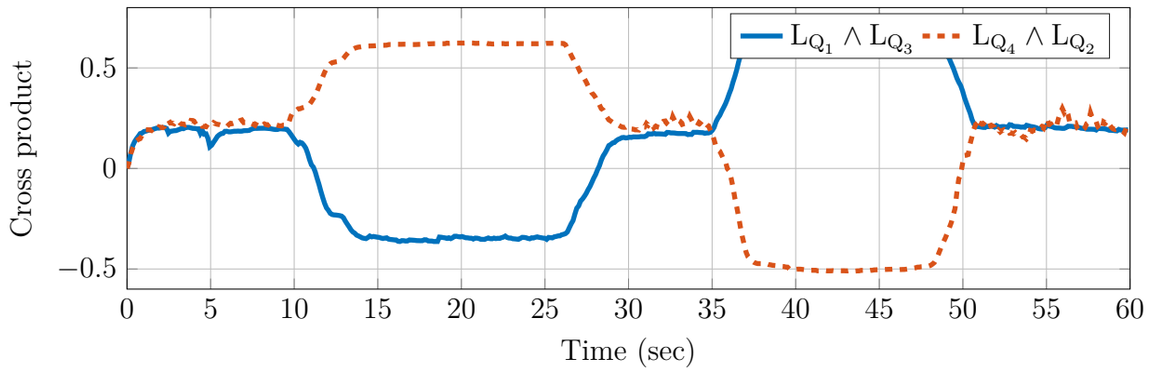
(b) Relative position

Figure 4.15 – (a) Cross product between principal lines of a corridor, (b) relative position between the cross product and the principal lines.

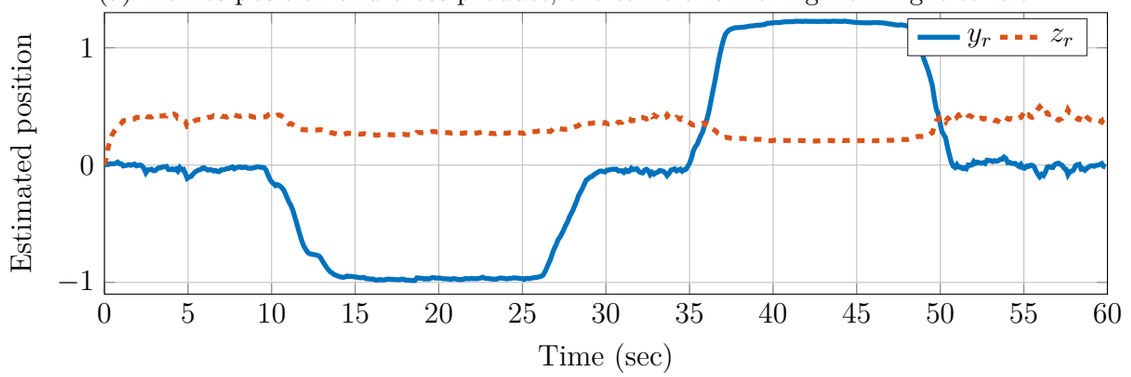
new value was estimated as can be observed at time 38s. The algorithm is only capable to estimate angles between ±10deg in roll and pitch, and ±30deg in yaw angle. It should be mentioned that this approach was thought to obtain the data of head direction in order to control it, which is the principal constraints to overcome the position algorithm.

4.3 Encountered problems

The camera pose algorithm was designed under certain restrictions. For example, the quadrotor counts with a low payload. The designed system was developed taking into consideration this problem, therefore the on-board camera is light and energy efficient. Meaning the image resolution is low (240x320 pixels), and sometimes it has a noisy reception. In addition, as the camera has only 30 fps, the pose algorithm has to be designed taking into account: the camera frames, computational time of the vision system, and the dynamic control of the system (quadrotor). In this section, some problems and some restrictions will be discussed.



(a) Frames position and cross product, the camera is moving from right to left

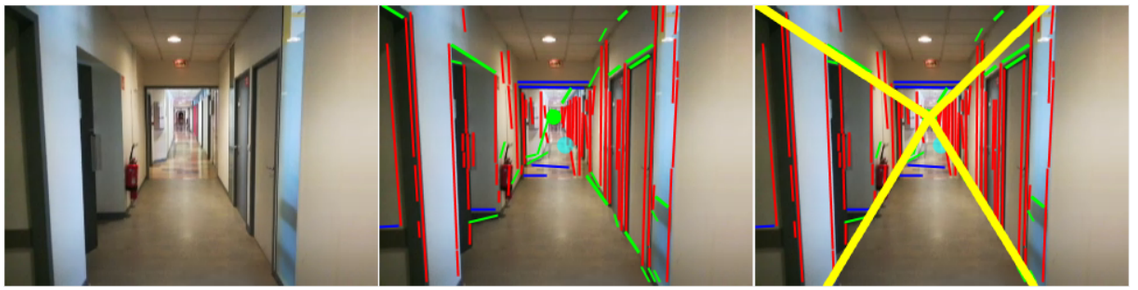


(b) Relative position

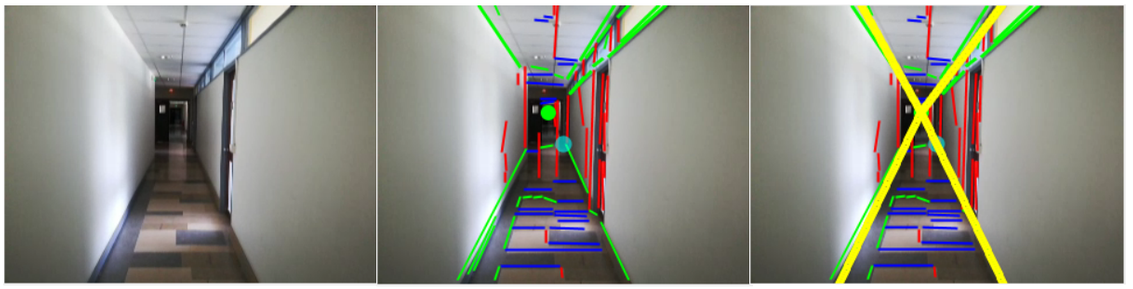
Figure 4.16 – (a) Cross product between principal lines of a corridor, (b) relative position between the cross product and the principal lines.

4.3.1 Camera motion

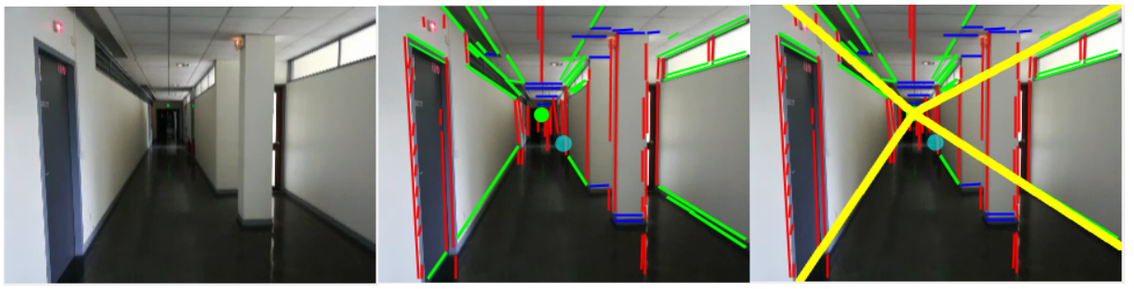
Nowadays, there are multiple cameras in the market that give us a professional image. The image quality is normally in HD and in some cases 4K, the video capture can reach up to 120 fps. However, the camera used in this work has a smaller dimensions than HD, and the frame frequency is 30 fps. There is a great difference between both cameras, yet a small camera was chosen in order to make an algorithm simple and fast. However, the image obtained by the small camera does not have sufficient quality, and sometimes the image obtained is blurred and noisy. Additionally, the camera motion is re-stringed by the rate frames. Then, if the velocity of the system is faster than the camera frames, the obtained images are blurred and sometimes also noisy, see Fig. 4.21a. These noisy images can affect the estimated data giving unwanted data in the vanishing point estimation and mainly poor principal lines, see Fig. 4.21b.



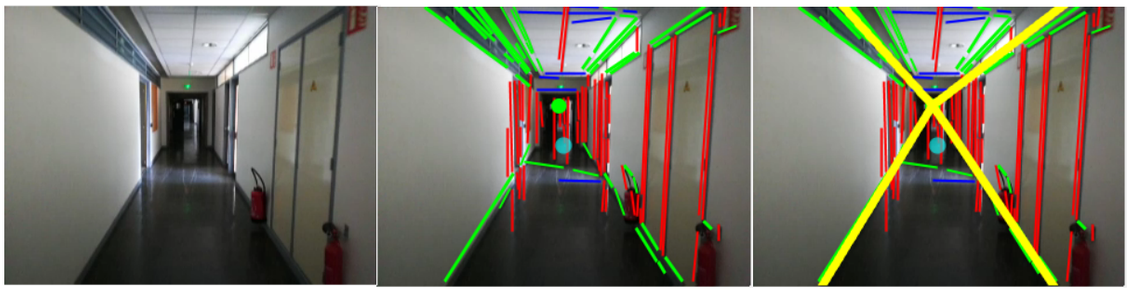
(a)



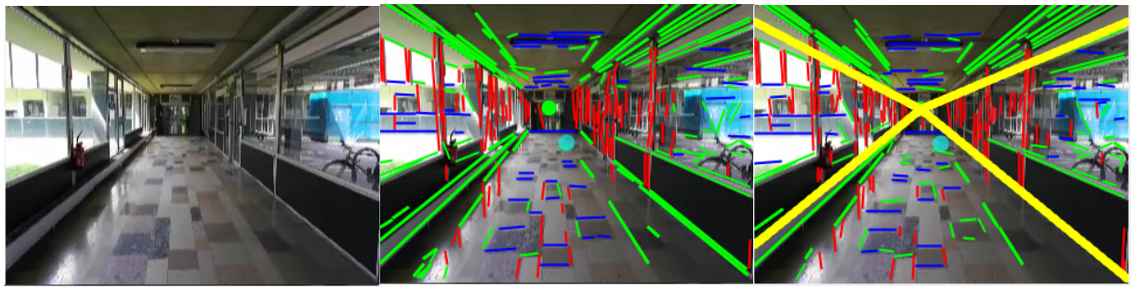
(b)



(c)



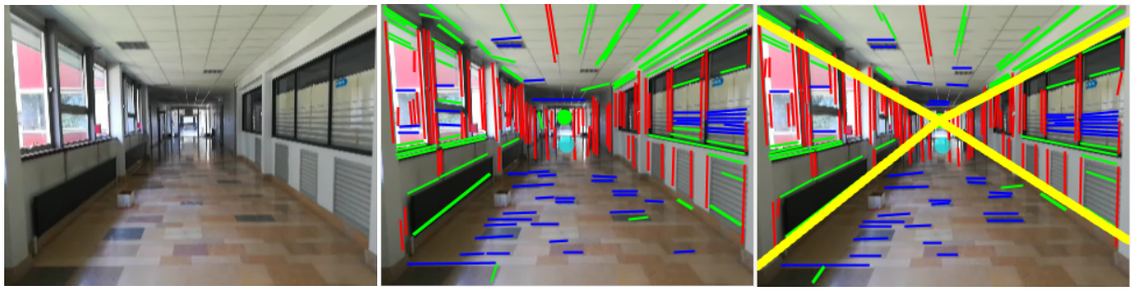
(d)



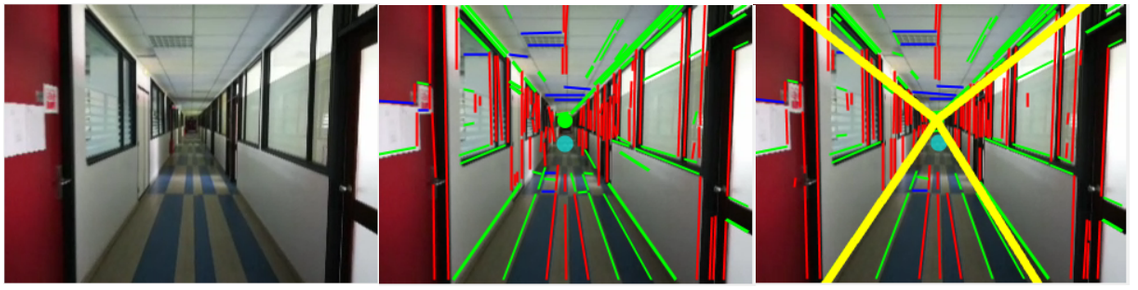
(e)

Figure 4.17 – Proposed algorithm applied to differents corridors

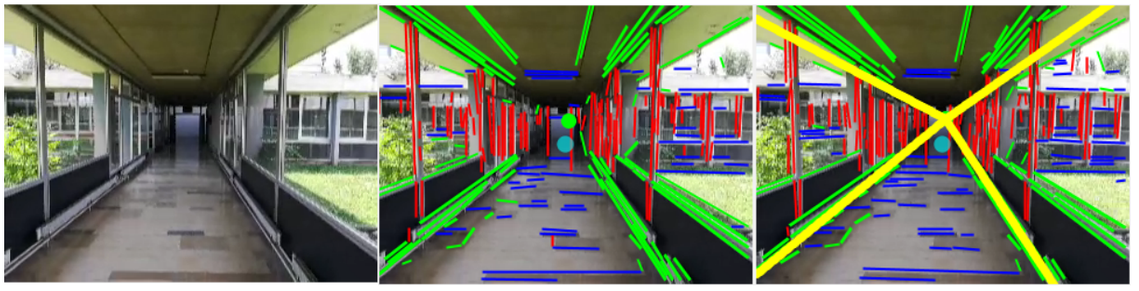
(a)



(b)



(c)



(d)

Figure 4.18 – Proposed algorithm applied to differents corridors (continuation).

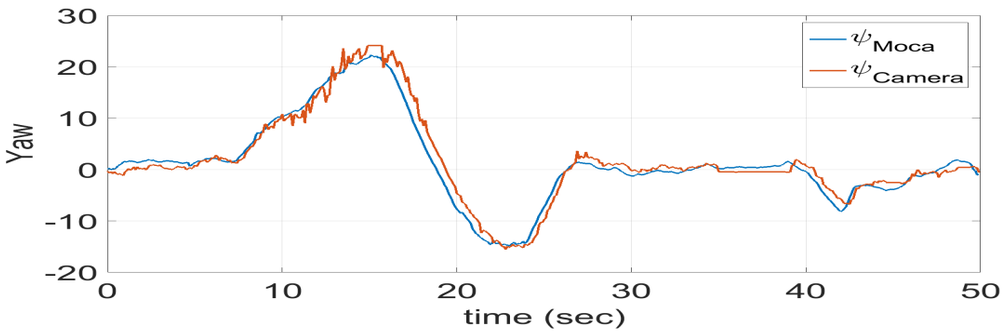
Figure 4.19 – A image, which was taken by a smartphone, was projected on wall using a commercial projector, then a video was taken by a smartphone whose some movements were made on yaw ψ, pitch θ and roll ϕ angles.

4.3.2 Video stream

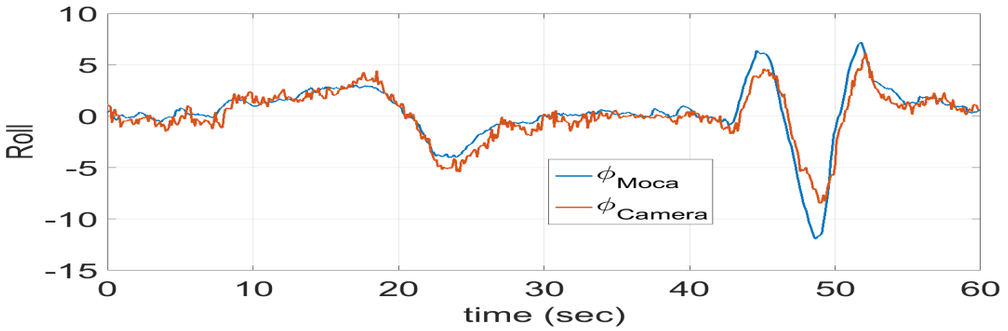
The obtained images can serve to take the streaming video which can give details about the environment. The image processing is desired to be made on-board; however, that requires a high computational cost. Thus, the proposed configuration is based in a commercial drone with an inexpensive camera, and the stream video is acquired via wireless using a FPV monitoring system 5.6 GHz. This information can be displayed directly in a monitor, in a FPV headset, or even data can be recovered on a PC using an interface video-to-USB giving the task to compute the pose algorithm in the ground station.

4.4 Summary

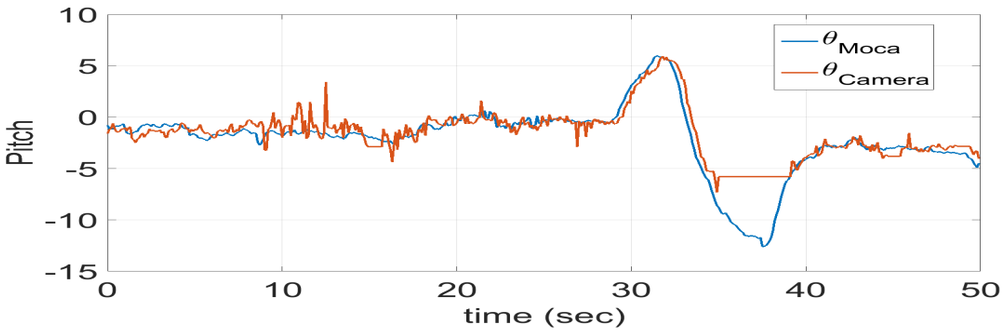
An algorithm to estimate the pose camera was described in this chapter which is based on camera calibration using vanishing points, and the collinearity of the edges lines described by a corridor. The camera calibration was obtained extracting one finite vanishing point and two infinite vanishing points where the camera is supposed to be always pointed to the end of the corridor. The relative position is obtained computing the collinearity of opposite edges lines. One can conclude that when the camera is not in the center of the corridor, the collinearity is different to zero. This value is used to obtain certain position with respect to the environment. At the end of chapter, some numerical tests were presented to see the behavior of the proposed solution.



(a)

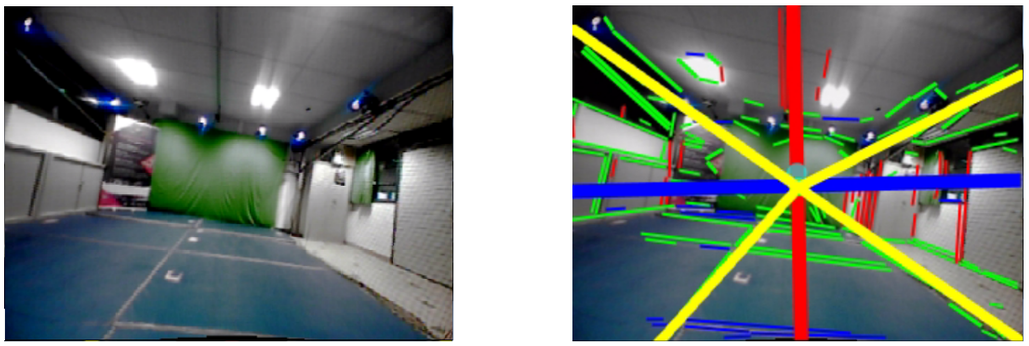


(b)



(c)

Figure 4.20 – Variation and comparison, moca vs camera, of angles: (a) yaw, (c)roll, and (c) pitch.



(a) (b)

Figure 4.21 – (a) Image has been taken by on-board camera when the drone was performing a sudden motion, (b) Lines were extracted during this motion, thus the estimated principal lines has been obtained with a fail due this sudden movement.

Chapter 5
Attitude and Position control

Merging visual information and control techniques, generate restrictions in control laws. Some kinds of restrictions like delays, noise and non-stability produces some errors in a feedback control. In Chapter 3 we described the quadrotor model which is composed of two dynamics model: attitude model and position model. Attitude control depends only of attitude dynamics model, but position control relies on attitude and position dynamics model. In this work we propose to use a visual system that add partial information to close the position control loop in position and attitude control for quadrotor head direction.

Vision system can be used to estimate certain position to close the loop of position control. Thus, the data obtained by the vision system presented in Chapter 4 can be used to control partially the position of the system. Moreover, head direction of the quadrotor system is controlled using the estimated data for the algorithm of camera calibration.

In this chapter we present a bounded attitude control 5.2. Then position control is explained in section 5.4. In section 5.5 is depicted some configurations proposed during the development of this work.

5.1 System description

The general control scheme of a quadrotor is shown in Fig. 5.1, where an internal loop and an external loop control can be observed. The internal loop comprises only the attitude control. The external loop control includes position and attitude controllers considering that the position control always depends of the attitude control. The mini quadrotor block has two outputs q and $\vec{\eta}$ which are respectively orientation and angular rates. $\vec{\varphi}$ and $\vec{r}$ represent the position system and the linear velocity. The attitude control block has seven inputs states $(q, \vec{\eta})$ and desired angles $[\phi_d, \theta_d, \psi_d]^T$, and three outputs $[\Gamma_\phi, \Gamma_\theta, \Gamma_\psi]^T$ being the torques computed by the attitude control. The position control yields the desired angles $[\phi_d, \theta_d, \psi_d]^T$ and total thrust F_u which are computed using the position states $\vec{\xi}$, signal

control $\vec{r}$ and rotation matrix ${}^{\mathbf{N}}R_{\mathbf{C}}$. Finally, the block titled as vision system represents the vision algorithm. The vector φ is partially estimated by the vision algorithm (y_r, z_r), and ${}^{\mathbf{N}}R_{\mathbf{C}}$ also estimated by vision system is employed to control the head-direction.

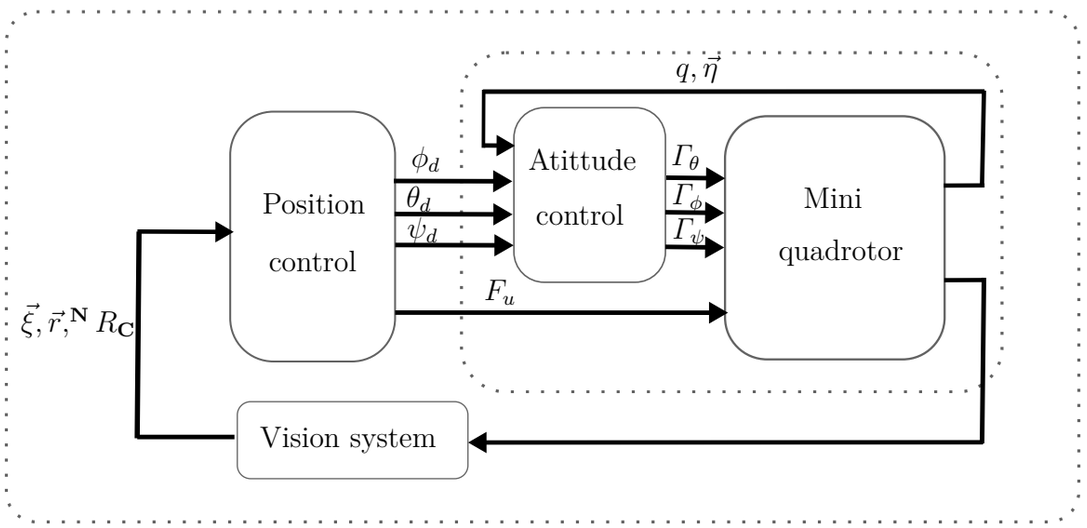


Figure 5.1 – Control scheme: the internal control loop controls the attitude while external control loop stabilizes the position.

5.2 Bounded control

In general, the proposed approach for the stabilization of the aerial vehicle consists on the development of a control law with bounded inputs. Since large amplitude disturbances can modify the system actuators into saturation threshold, the usage of this kind of algorithms becomes a good option for control design. As was shown, among others, in Bernstein and Michel [1995] many works were presented giving a good perspective of the control of systems with bounded inputs.

The works considering this kind of approach are mainly focused on the stabilization of linear systems, known as chains of integrators. Normally, these systems are represented by the form:

$$\dot{x} = Ax + Bu \tag{5.1}$$

where the state vector x is a column vector of length n, the input vector u is a column vector of length m, $A \in \mathbb{R}^{n \times n}$ depicts a square matrix with constant coefficients, and $B \in \mathbb{R}^n$ depicts a matrix with weight coefficients over inputs.

The control amplitude is limited by physical constraints in the system; in this case the physical constraints over the actuators:

$$-\bar{u} \leq u \leq \bar{u} \tag{5.2}$$

where $\bar{u}$ is a positive number representing the threshold of work for the control law, $-\bar{u}$ is negative saturation and u is positive saturation.

In literature, some approaches have been proposed. Some of them are: Optimal control, Pseudo-optimal control based on Ricatti equation, and the most popular for simplicity and regularity properties is non-linear control. This field of research has been initially proposed by Teel [1992]. Then, this result was generalized in Sussmann et al. [1994]. However, in these works, system performance in closed loop is affected in increasing dimensions. All of the aforesaid approaches of these techniques showed the interest on systems stabilization by bounded inputs in automatic control community. Nowadays, these strategies are highly implemented due to its simplicity and high performance.

5.3 Bounded attitude control

Considering the attitude dynamics of quadrotor Σ_R (5.3) and assuming that the angular positions are known, the attitude scheme control is depicted in Fig. 5.2. Then, the internal control can be computed with quaternion and angular velocities $(q, \vec{\eta})$. The desired angles $[\phi_d, \theta_d, \psi_d]^T$ and total thrust F_u are computed by the position control law, where $[\phi_d, \theta_d, \psi_d]^T$ are given by theirs equivalent in quaternions q_d.

$$\Sigma_R \quad : \quad \begin{cases} \dot{q} = \dfrac{1}{2}\Xi(q)\vec{\eta} \\ J\dot{\vec{\eta}} = -[\vec{\eta}^{\times}]J\vec{\eta} + \Gamma \end{cases} \tag{5.3}$$

5.3.1 Problem statement

The goal is to design a control law which drives the quadrotor to attitude stabilization. In other words, let q_d denoting the constant quadrotor stabilization orientation then

$$q \to q_d, \ \vec{\eta} \to [0, 0, 0]^T \text{ as } t \to \infty \tag{5.4}$$

The quaternion error that represents the attitude error between the current orientation. To drive the quadrotor to attitude stabilization, $q_d = [\pm 1 \ 0 \ 0 \ 0]^T$, the quaternion error coincides with the current attitude quaternion, that is, $q_e = q$. The control objective

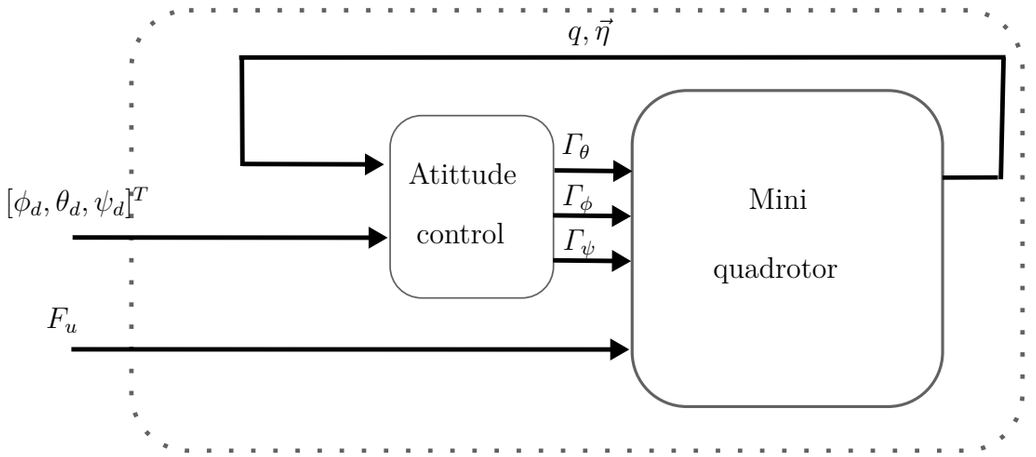


Figure 5.2 – Attitude control scheme.

is then

$$q \to [\pm 1\ 0\ 0\ 0]^T,\ \vec{\eta} \to [0,0,0]^T \text{ as } t \to \infty \tag{5.5}$$

Furthermore, it is known that the actuator saturation reduces the benefits of the feedback. When the controller continuously outputs infeasible control signals that saturates the actuators, system instability may follow. Then, besides the asymptotic stability, the control law also takes into account the physical constraints of the control system, in order to apply only feasible control signals to the actuators.

5.3.2 Attitude control

We will start with a definition to specify the type of control law obtained,

Theorem 5.3.1 (Guerrero-Castellanos et al. [2011b]) *Consider the rigid body rotational dynamics described by* (5.3) *with the following bounded control inputs* $\Gamma = [\Gamma_\phi, \Gamma_\theta, \Gamma_\psi]^T = [\Gamma_1, \Gamma_2, \Gamma_3]^T$*, whose subscript is changed to express the following equations, defined by:*

$$\Gamma_i = -\sigma_{\overline{\Gamma_i}}\left(\frac{k\vec{\eta_i}}{\rho_i} + sign(q_0)k\vec{q_i}\right) \tag{5.6}$$

$\overline{\Gamma}_i$ with $i \in 1,2,3$ represents the physical bound on the *i-th* torque Γ_i. q_0 represents the scalar part of the quaternion and $\vec{q_i}$ indicates the vector quaternion. $\vec{\eta_i}$ specifies the angular velocity. k marks a real parameter such that $0 < k \leq min_i\overline{\Gamma}_i/2$. ρ_i is a strictly positive real parameter. Then the inputs (5.6) asymptotically stabilize the rigid body to the origin $[1\ 0^T_{3\times1}0^T_{3\times1}]^T$ (*i.e.* $q_0 = 1, q_v = [0,0,0]^T$ and $\vec{\eta} = [0,0,0]^T$) with a domain of

attraction equal to $\mathbb{S}^3 \times \mathbb{R}^3 \setminus [1\ 0_{3\times1}^T 0_{3\times1}^T]^T$. In the case where the asymptotic condition $q \Rightarrow q_d$ with $q_d = [1, 0, 0, 0]^T$ is considered, the control law to be applied becomes

$$\Gamma_i = -\sigma_{\overline{\Gamma_i}}\left(\frac{k\vec{\eta_i}}{\rho_i} + sign(q_{e_0})k\vec{q_{e_i}}\right) \tag{5.7}$$

where q_e represents the attitude error between the current orientation and the desired one.The proof of stability is detailed in Guerrero-Castellanos et al. [2011b].

5.4 Position control

The position control design takes into consideration the dynamic position of Σ_T (5.8). The scheme control is depicted in Fig. 5.3. The position controller computes the angular position $[\phi_d, \theta_d, \psi_d]^T$ and the total thrust F_u according to the inputs references $[x_d, y_d, z_d]^T$.

$$\Sigma_T \ := \ \begin{cases} \dot{\vec{\xi}} = \vec{r} \\ m\dot{\vec{r}} = -mg\vec{\xi_3} + RF_u\vec{\xi_3} \end{cases} \tag{5.8}$$

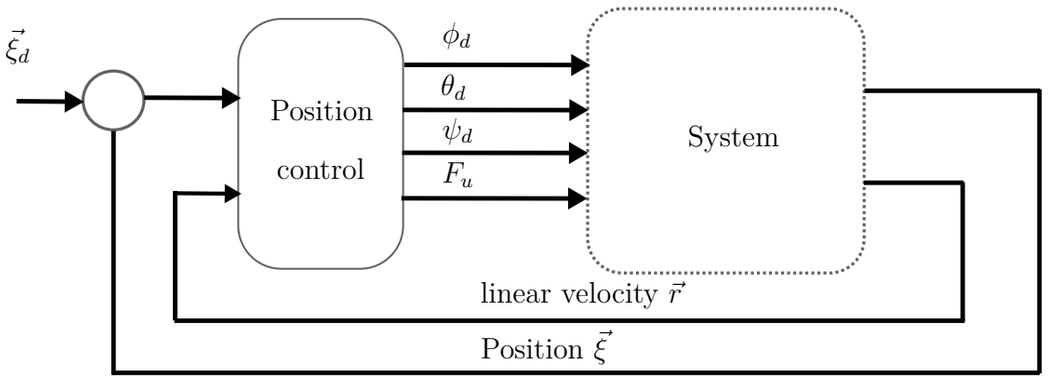


Figure 5.3 – Position control scheme.

5.4.1 Problem statement

The objective is to design a control law with the inner-outer loop configuration, which stabilizes the quadrotor to a desired position, from attitude stabilization problem solved. In other words, once the control law has stabilized the attitude of the system, $\lim_{t\to\infty}[q^T, \vec{\eta}^T]^T = [q_d^T, \vec{0}_{3\times1}^T]^T$, quadrotor desire position can be reached $\lim_{t\to\infty}[\vec{\xi}^T, \vec{r}^T] = [\vec{\xi}_d^T, \vec{0}_{3\times1}^T]^T$.

5.4.2 Design of position control

The dynamics of the whole system is obtained with the Newton-Euler formalism and the kinematics is represented using the quaternions formalism. This system can be seen as a cascade system, where the translational dynamics depends on the attitude, but the attitude dynamics does not depend on the translational one. Now, assume that using the control law (5.6) one can stabilize the yaw dynamics, that is $\psi = 0$, which is given by the head-direction obtained by vision system according to our configuration . Then after a sufficiently long time, system (5.8) becomes as follow

$$\begin{bmatrix} \dot{x}_n \\ \dot{y}_n \\ \dot{z}_n \end{bmatrix} = \begin{bmatrix} r_x \\ r_y \\ r_z \end{bmatrix}, \tag{5.9}$$

$$\begin{bmatrix} \dot{r}_x \\ \dot{r}_y \\ \dot{r}_z \end{bmatrix} = \begin{bmatrix} -\frac{F_u}{m}\sin\theta \\ \frac{F_u}{m}\sin\phi\cos\theta \\ \frac{F_u}{m}\cos\phi\cos\theta - g \end{bmatrix}, \tag{5.10}$$

With an appropriate choice of these targets configurations, it will be possible to transform (5.9) and (5.10) into three independents triple linear integrators. For this, take us

$$\begin{aligned} \phi_d &:= \arctan\left(\frac{\varsigma_2}{\varsigma_3 + g}\right), \\ \theta_d &:= \arcsin\left(\frac{-\varsigma_1}{\sqrt{\varsigma_1^2 + \varsigma_2^2 + (\varsigma_3 + g)^2}}\right) \end{aligned} \tag{5.11}$$

$$F_u = m\sqrt{\varsigma_1^2 + \varsigma_2^2 + (\varsigma_3 + g)^2} \tag{5.12}$$

where $\varrho = [\int x_n, x_n, r_x, \int y_n, y_n, r_y, \int z_n, z_n, r_z]^T = [\varrho_1, \varrho_2, \varrho_3, \varrho_4, \varrho_5, \varrho_6, \varrho_7, \varrho_8, \varrho_9]^T$, then (5.9)-(5.10) becomes:

$$\Sigma_x : \begin{cases} \dot{\varrho}_1 = \varrho_2 \\ \dot{\varrho}_2 = \varrho_3 \\ \dot{\varrho}_3 = \varsigma_1 \end{cases} \tag{5.13}$$

$$\Sigma_y : \begin{cases} \dot{\varrho}_4 = \varrho_5 \\ \dot{\varrho}_5 = \varrho_6 \\ \dot{\varrho}_6 = \varsigma_2 \end{cases} \tag{5.14}$$

$$\Sigma_z : \begin{cases} \dot{\varrho}_7 = \varrho_8 \\ \dot{\varrho}_8 = \varrho_9 \\ \dot{\varrho}_9 = \varsigma_3 \end{cases} \tag{5.15}$$

Since the chains of integrators given in (5.13)-(5.15) have the same form, a control law has been proposed in Cruz-José et al. [2012], and has been established by the next theorem:

Theorem 5.4.1 (Cruz-José et al.) *Consider the quadrotor translational dynamics expressed in (5.9-5.10). Then, the thrust input F_u with $\varsigma_1, \varsigma_2, \varsigma_3$ as in (5.16), where $\sigma_{M_1}(\cdot)$ is defined in (**??**) with $M_1 = 1$ and ς_i are given by (5.17), $a_{(1,2,3)}, b_{(1,2,3)}, c_{(1,2,3)} > 0$ tuning parameters such that $(a,b,c)_1 > (a,b,c)_2 + (a,b,c)_3$, $(a,b,c)_2 > (a,b,c)_3$, stabilizes globally and asymptotically the quadrotor translational dynamics at the origin. Furthermore, if none of the σ_{M_1} are saturated, the poles of the linearized closed-loop for the subsystems (5.13)-(5.15) reside at $-(a,b,c)_1, -(a,b,c)_2, -(a,b,c)_3$, respectively.*

$$\begin{aligned}
\varsigma_1 &:= -\vartheta_1\{a_3\sigma_{M_1}[\frac{1}{\vartheta_1}(a_2\varrho_1 + \varrho_2 + \varrho_3)] + a_2\sigma_{M_1}[\frac{1}{\vartheta_1}(a_1\varrho_2 + \varrho_3)] + a_1\sigma_{M_1}[\frac{1}{\vartheta_1}(\varrho_3)]\} \\
\varsigma_2 &:= -\vartheta_2\{b_3\sigma_{M_1}[\frac{1}{\vartheta_2}(b_2\varrho_4 + \varrho_5 + \varrho_6)] + b_2\sigma_{M_1}[\frac{1}{\vartheta_2}(b_1\varrho_5 + \varrho_6)] + b_1\sigma_{M_1}[\frac{1}{\vartheta_2}(\varrho_6)]\} \\
\varsigma_3 &:= -\vartheta_3\{c_3\sigma_{M_1}[\frac{1}{\vartheta_3}(c_2\varrho_7 + \varrho_8 + \varrho_9)] + c_2\sigma_{M_1}[\frac{1}{\vartheta_3}(c_1\varrho_8 + \varrho_9)] + c_1\sigma_{M_1}[\frac{1}{\vartheta_3}(\varrho_9)]\}
\end{aligned} \tag{5.16}$$

$$\begin{aligned}
\vartheta_1 &= \bar{\varsigma}_1/(a_1 + a_2 + a_3), \\
\vartheta_2 &= \bar{\varsigma}_2/(b_1 + b_2 + b_3), \\
\vartheta_3 &= \bar{\varsigma}_3/(c_1 + c_2 + c_3)
\end{aligned} \tag{5.17}$$

Then, the control laws in (5.16) exponentially stabilize the systems (5.9)-(5.10) to the desired position $(\varrho_1, \varrho_2) = (\varrho_{dx}, 0)$, $(\varrho_3, \varrho_4) = (\varrho_{dy}, 0)$ and $(\varrho_5, \varrho_6) = (\varrho_{dz}, 0)$.

Remark 5.4.2 *In the above theorem, the stabilization goal is the origin. In the case where the asymptotic condition is different from the origin, the variables $\varrho_2, \varrho_5, \varrho_8$ should be replaced in the control law (5.16) by $e_1 = \varrho_2 - \varrho_x^d$, $e_2 = \varrho_5 - \varrho_y^d$, $e_3 = \varrho_8 - \varrho_z^d$, respectively. In this case $\varrho_x^d, \varrho_y^d, \varrho_z^d$ represent the desired position in the space.*

Vision algorithm gives the relative position from y-axis and z-axis, called y_r and z_r in section 4.2.5, and the position over x-axis is controlled using measures from the motion capture system, or the desired values for x_d could be given by a pilot.

5.5 System navigation configuration

Overview of the system is depicted in Fig. 5.4-5.6, where three configurations can be observed: the complete simulation, the quasi-virtual system and the real video scheme. All configurations have a function switch which is used to commute the real data and the estimated data.

Fig. 5.4 shows the simulation strategy. The scheme is separated in two parts: internal control loop and external control loop. The internal control loop takes into account the system model (quadrotor), the attitude control, and the external control loop includes the position control and the computer vision. The operation is as follows:

- The system model yields the states of the system: position vector $\vec{\varphi}$, linear velocity vector $\vec{r}$, rotation matrix R and angular rate $\vec{\eta}$.
- These data are the inputs to the Babylon 3D system Catuhe and Rousset [2016], which is a 3D engine based on webGL and Javascript, and at the same time this information is applied to the function switch to obtain the inputs errors.
- The 3D engines come up with the image video serving as input to the image processing.
- The vision algorithm estimates position in y and z axes and the rotation matrix R.
- Function switch performs a selection in both model real data and estimated data and computes the vector error $\vec{e} = \vec{\varphi}_{ref} - \vec{\varphi}_r$ taking into account the reference position vector $\vec{\varphi}_{ref}$, and the real position vector $\vec{\varphi}_r$.
- From $\vec{e}$, position control $\varsigma_{1,2,3}$ is calculated. Finally a conversion between the position control to the desired angles $(\phi_d, \theta_d, \psi_d)$ and the thrust F_u is accomplished .

5.5.1 Simulation setup

Even though the performance of this configuration is good, this approach is a complete simulation without any external disturbance. Then, a quasi-virtual system is proposed in Fig. 5.5, which is the combination between a real system and virtual image. In this scheme, a real system (mini quadrotor) and a virtual image created by Babylon 3D are

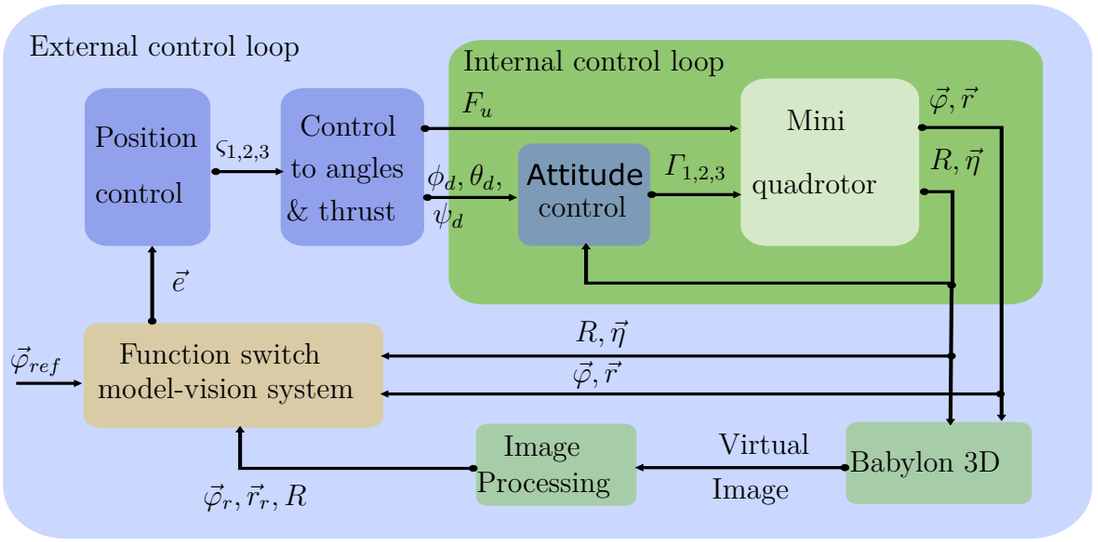


Figure 5.4 – Quadrotor system navigation strategy: Simulation scheme.

used. The running is similar to the first configuration, and the results are presented in the next chapter.

Fig. 5.6 depicts a configuration using a system where the image video is acquired in real time. The video is obtained using the embedded camera over the quadrotor where the streaming video is used as input to run the block of vision system.

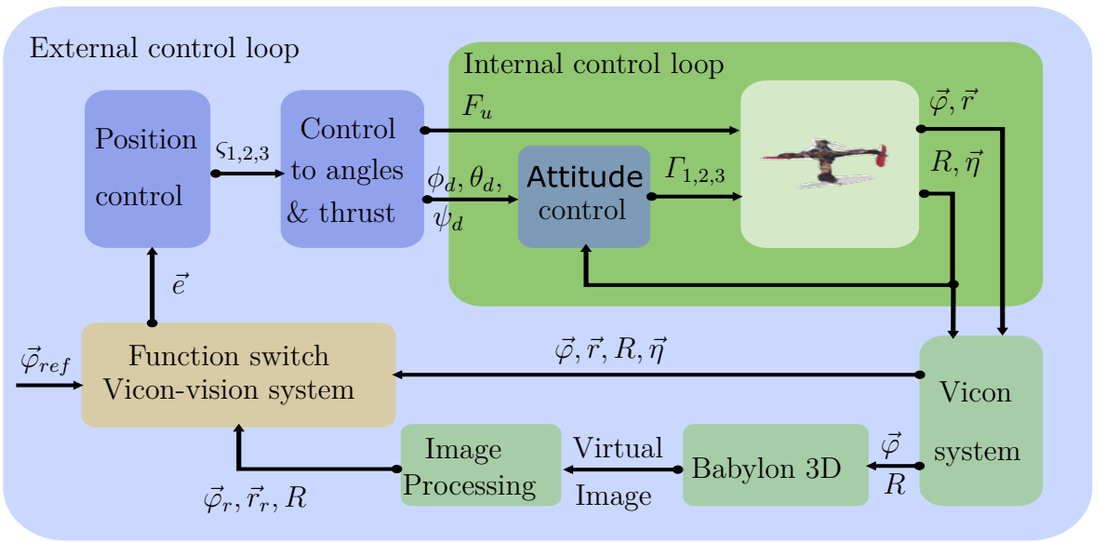


Figure 5.5 – Quadrotor system navigation strategy: Quasi virtual scenario.

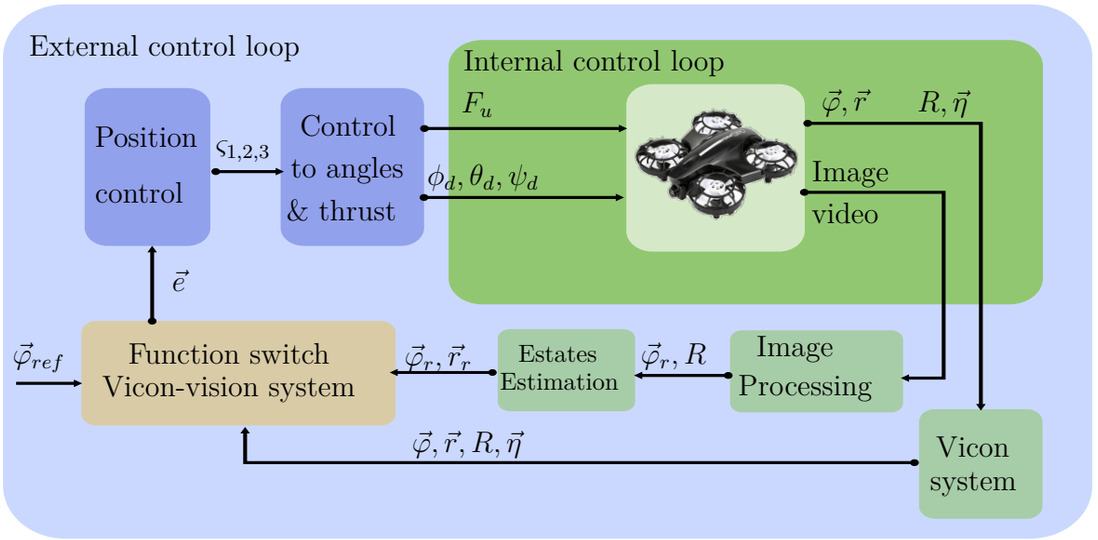


Figure 5.6 – Quadrotor system stabilization strategy: Real video scheme.

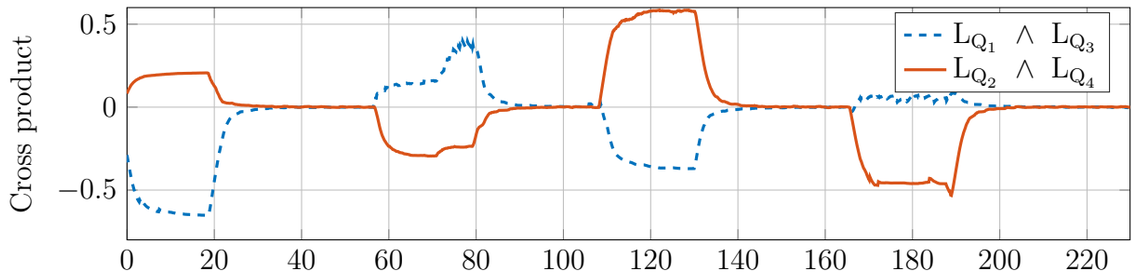
5.6 Simulation tests

Simulation tests were carried out for validating the vision algorithm presented in Chapter 3. In order to simplify the simulation tests, the x coordinate and the pitch movement θ are fixed for limitations in the workspace. Therefore, the quadrotor will move as a PVTOL aircraft evolving in the $z-y$ plane, and the ψ angle will take the vanishing point direction (head-direction) having a value near to zero. However, the x-axis is stabilized using measurements coming from Vicon system with this data input quadrotor evolves in hover state in the middle of virtual corridor at Vicon position $x = 0$m, $y = 0$m and $z = 0.9$m.

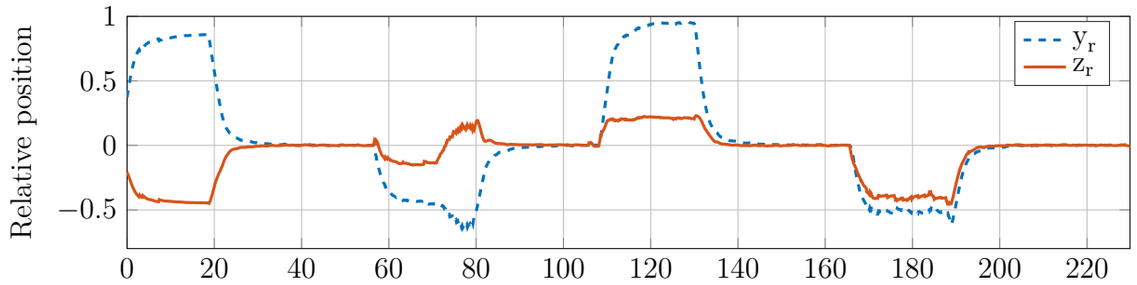
From this validation steps are the following: 1) quadrotor is placed at different positions in the y and z coordinates at different times, using states from model system (simulator) or using Vicon measurements called in this work: quasi-virtual system, more details in next Chapter 6. 2) measurements coming from simulator or quasi-virtual system (only for y and z) are switched with related position (y_r, z_r) calculated with our proposed vision algorithm. Our goal is to keep the vehicle in the middle of the corridor, for that control laws are computed using the (y_r, z_r) measurements in order to place our quadrotor system in a desired position.

5.6.1 Analysis results

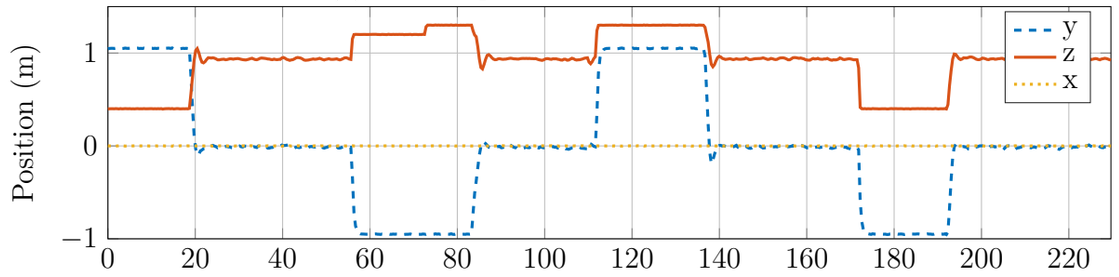
This test is carried out using two sessions of Matlab. The first one is used to run the complete simulation of the vehicle model, orientation and position control. The other one



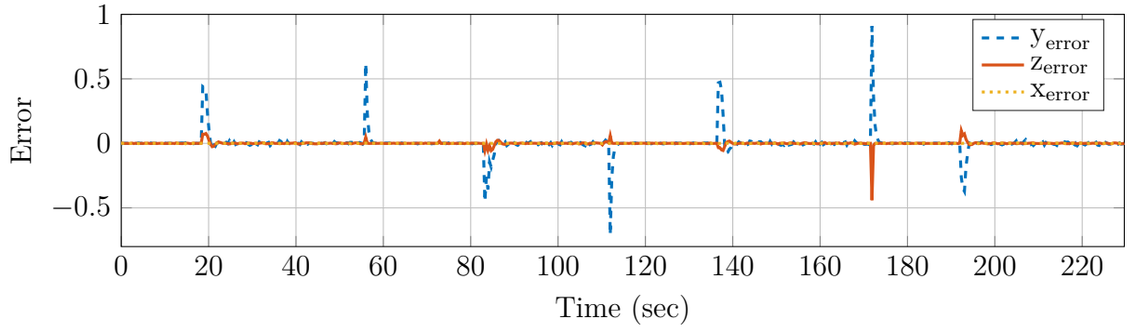
(a) Cross product of principal lines



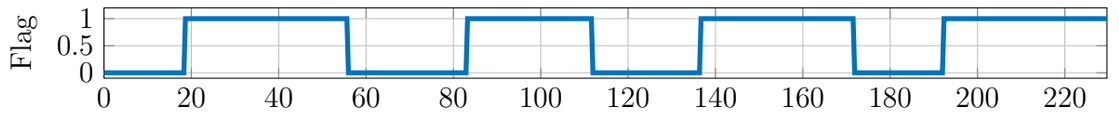
(b) Relative position



(c) Simulator position



(d) Error



(e) 0 - Vicon measurements, 1 - Estimated vision variables

Figure 5.7 – Stabilization using model simulator and Babylon 3D.

Table 5.1 – Position to simulation tests

Position	1	2	3	4	5	6	7	8
Time (s)	0	17	53	80	120	180	170	190
y	1.0	0	-1.0	0	1.0	0	-1.0	0.0
z	0.5	0.95	1.3	0.95	1.3	0.95	0.5	0.95

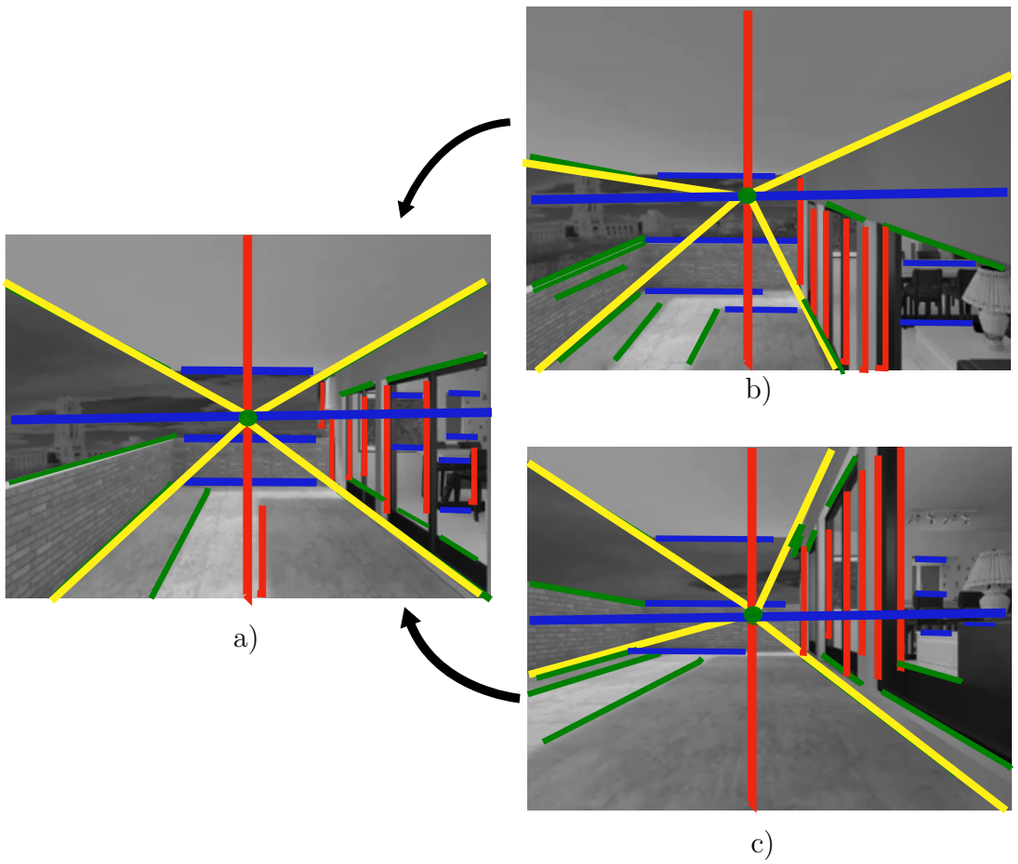


Figure 5.8 – Images (a), (b) and (c) represent respectively positions 4 , 3 and 7 of table 5.1.

runs the engine BabylonJs and the vision system. The communication between these two sessions was done via UDP (User Datagram Protocol). The acquisition of the image video is at 200 hertz and the vision algorithm runs at 22 hertz.

Fig. 5.7 shows the numerical behavior of the system applying the sequence from the Table 5.1. Firstly, Fig. 5.7e displays the flag which indicates changes of the different positions. Here, values indicates the moment when the estimated states are used 1 means

real states from simulator and 0 means estimated states from vision system. Fig. 5.7a depicts the cross product of the main lines according to the equations (4.18) and (4.19), and it can be seen the behavior of this product with respect to collinearity between lines. Besides, if lines are collinear cross product is equal to zero (that can be seen in positions $2, 4, 6$ and 8 in Table 5.1).

The real position from the model system is represented in Fig. 5.7c. The variation of y and z positions are given according to Table 5.1, notice that x position always remains constant. The variations of estimated position over y and z axes are shown in Fig. 5.7b whose values are obtained using the data of the Fig. 5.7a and equations (4.20)-(4.21). The behavior over y-axis is similar in both figures Fig. 5.7b and Fig. 5.7c. However, the z real position from Fig. 5.7c is quite different from the estimated position in z-axis Fig. 5.7b due to the change of the origin from camera system. Although the data plotted in Fig. 5.7c are given in meters, the metric unit in Fig. 5.7b is not similar; though, this information is used as the position error to the origin defined by the vision algorithm. Finally, in Fig. 5.7d the error evolution is shown, where it can be seen the moment when the estimated values are switched.

Fig. 5.8 depicts some images of the sequence video. Fig. 5.8c shows the position 7 of Table 5.1 which represents the initial condition of the virtual quadrotor system. Position control is applied to converge to the center of the corridor, Fig. 5.8a. The same case can be seen in Fig. 5.8b. It can be seen clearly that vanishing point is moved according to the orientation and camera position. Moreover, the tilt variation in lines, which converge to the vp, are used to extract certain position of the camera. This information is used to feedback the position control and centering the virtual quadrotor to the center of the virtual corridor.

5.7 Summary

In this chapter, a bounded attitude and position controllers were detailed to be implemented in our quadrotor platform. Then, a bounded position control was presented. Since vision system is only able to estimate position over (y_r, z_r) axes, position control was applied to y_n, z_n axes in real world. x_n-axis could be commanded either closing the loop control from data obtained by Vicon system or by a remote control with a pilot.

Three configurations for the complete system were proposed. The first one is a total simulation, the second one is a combination of virtual image (Babylon 3D) and a real quadrotor, quasi-virtual system, and the third one is a scheme using a real video and a real quadrotor. Finally, results obtained from simulation were shown to validate the proposed algorithm.

Chapter 6
Platform and experimental validation

In this chapter we present the experimental set up, including the MOCA room at GIPSA-lab, ground station, position estimation system and aerial platforms developed during my thesis, as well as to introduce the experimental results and subsequently, theirs analysis.

First, to implement the position control law, the position of the system must be known, for this, the MOCA room and vision system were used. The first one is composed of 12 cameras and a ground station, which allows the implementation of the algorithms through MATLAB/Simulink and sends them to the system through radio signals. More details about this system are provided in this chapter. The second one is the vision system proposed in this work, composed of a single on-board camera into the quadrotor vehicle, and a PC receiving the streaming video to estimate the position with respect to the geometry of a corridor.

During my thesis, different platforms were developed and used to validate our proposed algorithm. First, a Flexbot micro quadrotor was used to test the attitude and position control laws. The general performance of this model was suitable; however, when the camera system was added to test the vision algorithm, the platform was not able to take off due to non powerful motors. Thus, a bigger platform was designed. The second platform was the Flexbot micro-hexacopter. The technical specifications of on-board flight controller and motors were similar to the first one, but this model also offered two extra actuators and a bigger battery. With this prototype, new experimental tests were performed, and the behavior of system was improved. The motors used with this commercial platforms were DC motors and its operational life is not so long, therefore, the performance of these ones were reduced drastically with each test.

Therefore, due to aforesaid problems. Two different frames were designed and 3D printed. The on-board flight controller was changed, with better specifications in terms of

processor and general performance. The DC motors were changed to brush-less motors, to increase the power and carrying capacity. Consequently, the use of speed controllers for the motors were needed. The propellers were enlarged according to the specifications of the motors as well as the size of the battery. The camera was added manually; however, the video stream was noisy, and sometimes the streaming video was lost. Thus, a commercial drone, see section 6.2.3, with an embedded camera was used in order to fix the streaming video problems. This quadrotor system has small size brush-less motors.

All the platforms are described in the next subsections, some characteristics of each one are given, but only the last two, will be detailed. At the end of chapter, experimental results are presented.

6.1 Moca Room and ground station

In order to test and to compare the data obtained by the proposed algorithm with respect to the real data acquired by the Moca system, a complete platform was developed. This stage is integrated with a motion capture (Vicon system) and a ground station. The Vicon system is used to obtain the real orientation and position of the system, and the ground station is concerned with calculating the position control, and the vision system.

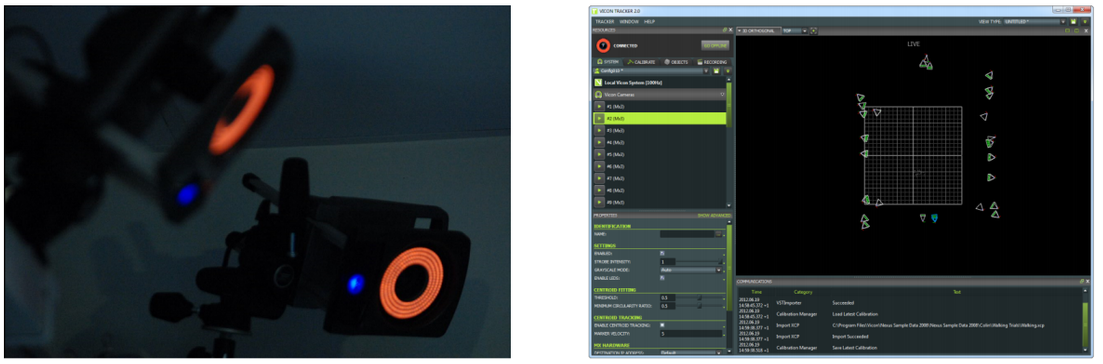
6.1.1 Moca Room

The motion acquisition is made through infrared cameras with emitters and receivers of infrared light and also through reflecting markers attached to the objects or individuals which can move inside the workspace. The MOCA room is composed of 12 VICON© cameras (T40 series), attached to a metal structure upward and pointing their vision towards a common area. There are also 8 digital cameras pointing to the same area, but these ones are used for objects reconstruction or motion capture by image processing. With this system, it is possible to compute the position and attitude up to 100Hz. Fig. 6.1 shows an image of the MOCA room and the reflecting markers.

A VICON camera is an infrared camera, which emits and receives infrared rays. A set of cameras pointing towards a common area is able to detect a reflective marker. The markers are little balls of retro-reflecting materials going from 0.5cm to 2cm of diameter. The cameras emit a very special light which makes the receivers sensitive only to this one, when a marker is placed into the area covered by the cameras, it creates a single point in the plane of each one of the cameras (if the area is well covered). Then, the information is collected in a computer running the VICON© tracker software. Fig. 6.2 shows an image of the used VICON cameras and the VICON tracker environment.

(a) (b)

Figure 6.1 – (a)MOCA room and (b)reflecting markers.



(a) (b)

Figure 6.2 – (a) VICON cameras and (b) VICON tracker environment.

6.1.2 Ground station

The ground station is composed by three computers: the first one is under the real time MATLAB/Simulink© environment. The second one is running the xPC-target tool- box, and also interfaces with the radio-frequency emitter and Vicon system. The Vicon system yields the states of the system, which are sent to MATLAB/Simulink through an UDP frame every 2ms. The position control algorithm is computed and implemented in real time at 200Hz on the xPC-Target. This PC also manages communications between the host and target PC, as well as the different inputs/outputs of the real-time application. The control variables are finally sent back to the system through GIPSA-lab's built-in bridge that converts UDP frames to DSM2 protocol. For this, the radio-frequency emitter is used. The third PC is used to compute the proposed algorithm using as input the streaming video acquired by the on-board camera. This video is transmitted to the PC using a video monitors on 5.8GHz. The estimated data (position and head-direction) are sent to the xPC-target via UDP frames. These estimated data are switched with the

data estimated by Vicon system in accordance to the experimental test. Fig. 6.3 shows a general overview of the computing process.

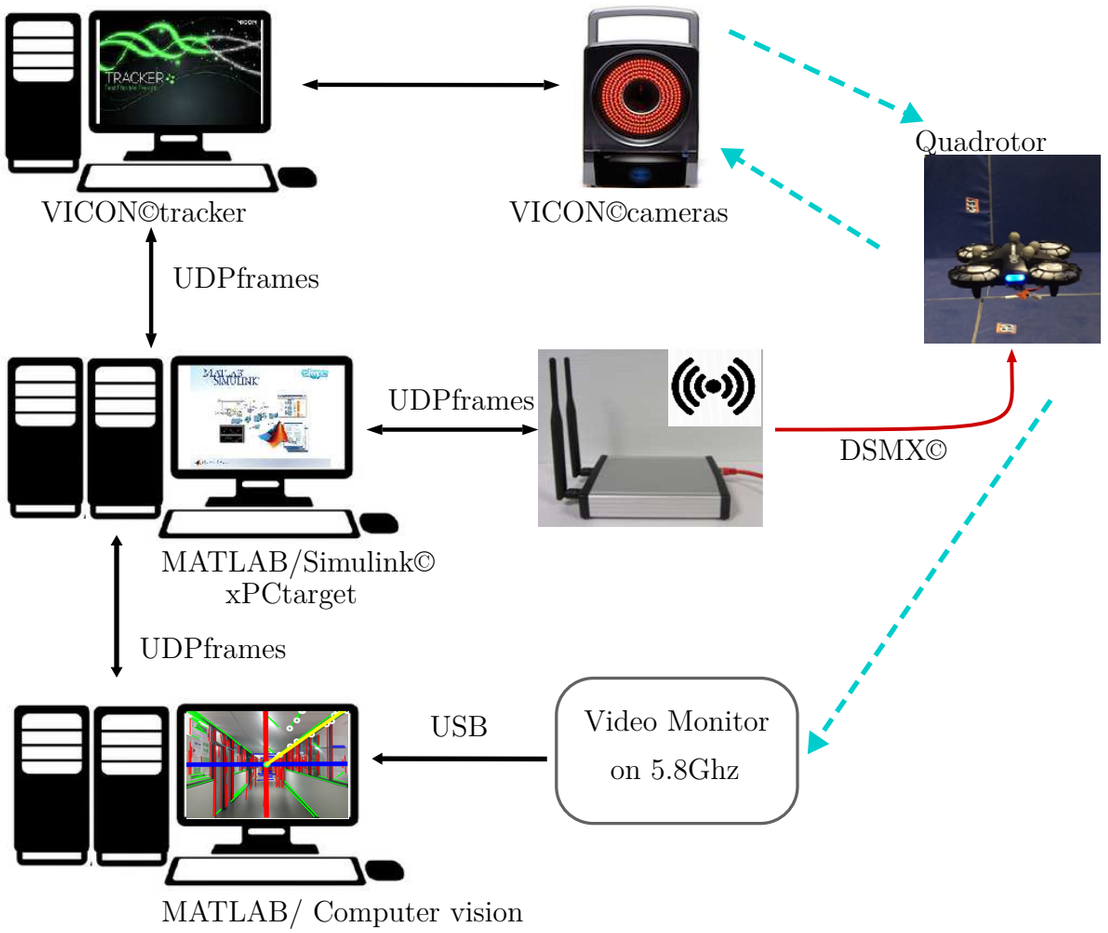


Figure 6.3 – Quadrotor control system process at MOCA room.

6.2 Experimental platforms

For the development of this work, 5 experimental platforms were used: 2 FLEXBOT prototypes, 2 home-made platforms and 1 Inductrix 200 FPV BNF.

6.2.1 Flexbot on the project

Two original flexbot prototypes and a tuned flexbot prototype were used in the past of the project. First, the flexbot quadrotor was used to test the orientation and position control laws, but due to its small size the vehicle was not able to fix a small camera. From

here, it was decided to move to the Flexbot hexa-copter, which has two extra actuators and propellers. However, batteries constraints reduced flight time were encountered.

For more information about the the FLEXBOT prototypes and the other platforms developed, please refer to Appendix A.

6.2.2 Home-made prototype: Mosca quadrotor

This aerial system consists of a home-made quadrotor, named " Mosca quadrotor". Two structures were designed and built to test our algorithms. The characteristics and parameters of the quadrotor are described in Table 6.1. The total weight of the quadrotor is about 280g and its carrying capacity is about 80g. The system is depicted in Fig. 6.4.

System	Description	Value	Units
	Mass (m)	280	g
	Distance (d)	10.7	cm
	Battery	7.4	V
	Carrying capacity	80	g
Quadrotor	Inertial moment x (J_ϕ)	0.0056	$Kg \cdot m^2$
	Inertial moment y (J_θ)	0.0056	$Kg \cdot m^2$
	Inertial moment z (J_ψ)	0.0087	$Kg \cdot m^2$
	Proportionality constant (b)	2615.23	N/s
	Proportionality constant (k)	257.80	N/s

Table 6.1 – Characteristics and parameters of Mosca quadrotor.

Figure 6.4 – Mosca quadrotor developed at GIPSA-LAB.

Flight controller

The attitude control law, see (5.6), for the quadrotor was programmed on the Copter board, which features an IMU sensor (MPU6050), integrated by gyros and accelerometers, the HMC5883L as 3-axis digital magnetometer, a MS5611-01BA03 high precision altimeter and the ATMega 2560-16AU as processor. The processor consists of a high-performance, low-power Atmel 8-bit AVR RISC-based micro-controller and it combines 256KB ISP flash memory, 8KB SRAM, 4KB EEPROM, 86 general purpose I/O lines, 32 general purpose working registers, real time counter, six flexible timer/counters with compare modes, PPM (Pulse Position Modulation), 4 USARTs, byte oriented 2-wire serial interface, 16-channel 10-bit A/D converter, and a JTAG interface for on-chip debugging. The device achieves a throughput of 16 MIPS at 16 MHz and operates between 4.5-5.5 volts.

The board dimensions are of $50 \times 50 \times 11.66mm$ and the weight of $14.5g$. The motors are connected to the card though the speed controllers (ESC). Fig. 6.5 shows the CRIUS flight controller board.

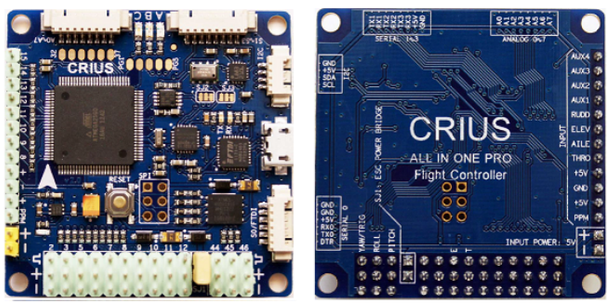


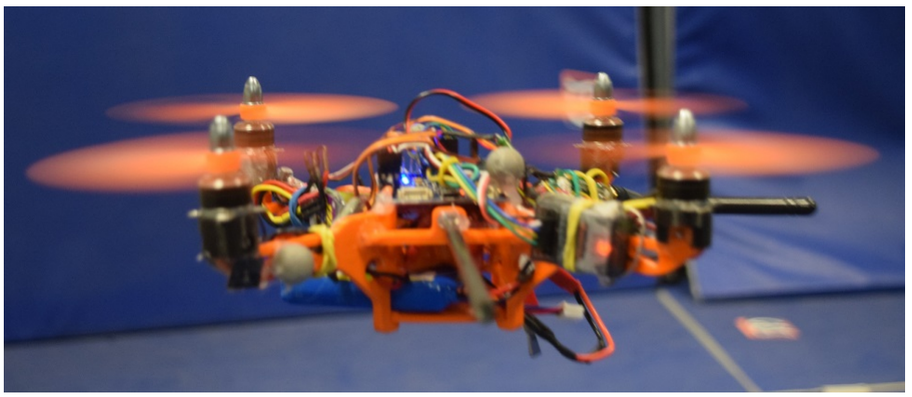
Figure 6.5 – CRIUS flight controller board.

Communication

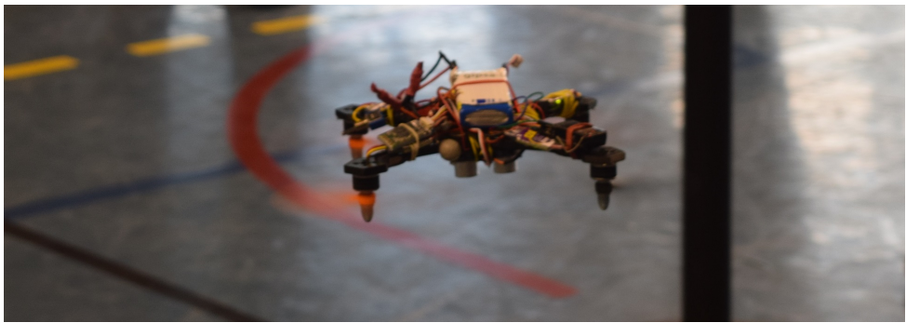
In order to communicate between the ground station and the platform, a Spektrum© AR6115e module was used. This one features a 2.4GHz 6-channel park flyer receiver with red led indicator under the DSM2/DSMX© protocol.

Frame

Two different frames were designed and 3D printed. Both systems are shown in Fig. 6.6a and Fig. 6.6b.



(a) Mosca robot version 1.0.



(b) Mosca robot version 2.0 (Inverted motor).

Figure 6.6 – Prototypes developed in GIPSA-LAB 2017.

6.2.3 Commercial prototype: Inductrix 200

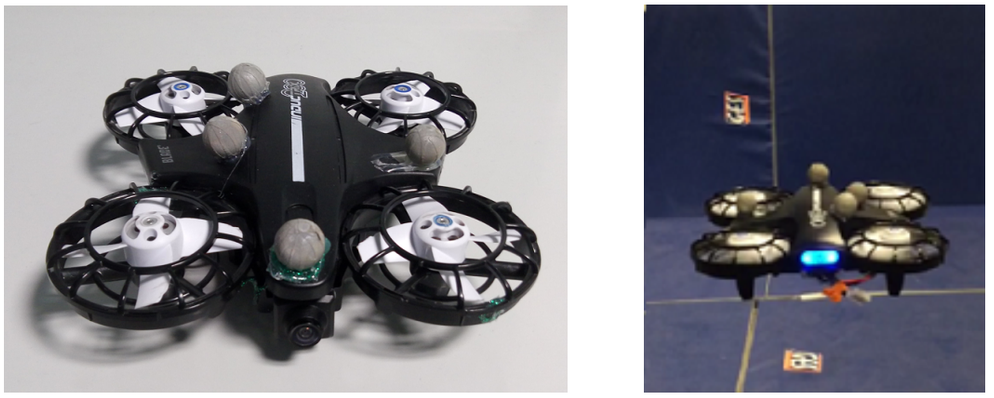
The Blade ©Inductrix™200 FPV quadrotor (Fig. 6.7) is a drone which has First Person View (FPV) capabilities in a compact aircraft, and is easy to fly. The FPV camera integrated into the lightweight airframe operates with a wide range of view screens and headsets. Since the Inductrix 200 FPV drone can give the immerse experience of riding air, this system has some advantages for the purposes of this thesis. The prototype has a lightweight, fully assembled airframe, and a fully integrated FPV flight camera compatible with FatShark headsets or video monitors at 5.8GHz. At the same time, the drone is compatible to Full-range 5+ channel, multi-function transmitter with Spektrum 2.4GHz DSM2/DSMX technology, and it is easy to communicate to the computer featuring the xPC-target system. The main characteristics are listed in Table 6.2.

6.2.4 Battery effects and motor speed control

A common problem experienced with quadrotors is the time-variant thrust response due to a drop of the battery voltage. In other words, for a same command received by the flight controller, the resulting thrust given by the motors will depend on the battery state

System	Description	Value	Units
	Mass (m)	185	g
	Length	15.5	cm
	Battery	$3S$ $800mAh$	LiPo

Table 6.2 – Characteristics and parameters of Inductrix 200.



(a) (b)

Figure 6.7 – a) Inductrix™200 FPV quadrotor, b) quadrotor in flying.

of charge. This time-variant response of the quadrotor requires a high integrator gain in the position controller which results in adding too much phase. For the Flexbot, the DC motors are controlled directly by the flight controller card through some transistors, but for the actual quadrotor, as power is required for the motors, Electronic Speed Controllers drivers (ESC) are used in order to provide the necessary amount of power and to handle the 3-phases of the motors. However, most of the ESCs do not achieve closed-loop control of the motor speed, and are then sensitive to the voltage drop of the battery. BLHeli is an open source project intended for replacing the official firmware of different ESCs drivers. The main advantage is that it provides a sensor-less closed-loop control mode of the motor speed. Therefore, the rotation speed should not be impacted by the battery's state of charge. Several adjustable parameters are available; however, finding the good values for our setup is quite difficult without any objective measurement. For this reason, a test bench for the couple motor/ESC driver has been built in order to quantify the effect of the different tuning parameters for a PI controller.

Motor test bench setup

We desired to acquire motor speed in order to measure its response time and to validate if the closed-loop control is able to reject a drop voltage. To measure speed motor a hall effect sensor is used, for considered motor model, here poles are directly visible by the

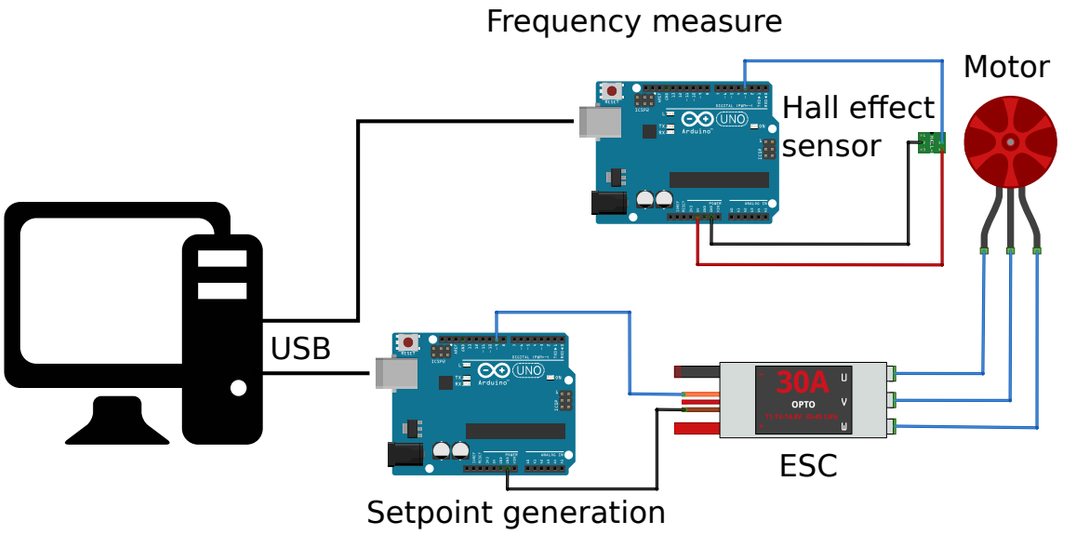


Figure 6.8 – Test bench ESC driver + motor.

sensor. An arduino Uno microcontroller has been programmed to compute the frequency given by the hall effect sensor, this frequency is then sent to a PC periodically. The PC sends several set-points through USB to another arduino card which generates the PPM signal for the ESC driver, coded like 1ms pulses translate to zero throttle, 2ms pulses coded to full throttle, Fig. 6.9 shows the input profile sent to the ESC driver, the first part (from 0 to 9 seconds) consists on an ascending and descending ramp, the second part (from 10 to 19 seconds) tests several step responses, the third part (from 20 to 22 seconds) consists on a high frequency reference, and during the last part, the voltage of the power supply is dropped by 1 Volt. A Lab-view real-time software has been built to send, receive and save the different data. Fig. 6.9 shows an overview of the motor test bench.

Results

18 configurations of PI control parameters have been tested. Fig. 6.10 shows the measured speed of the motor for 4 different tuning parameters. It can be seen that the response to the ramp input is the same for all the closed-loop parameters. To determine the best values for the proportional gain (Kp) and the integral gain (Ki), two important behaviors are considered : the rejection of a voltage drop and the step response. Fig. 6.11 shows a detailed zoom around two regions of interest:

- It can be seen clearly on Fig. 6.11a that a voltage drop equal to 1 Volt (near 24 s) results in a speed drop of about 15 Hz for the open-loop control, whereas all the closed-loop response are less impacted and tend to recover the speed drop. As the

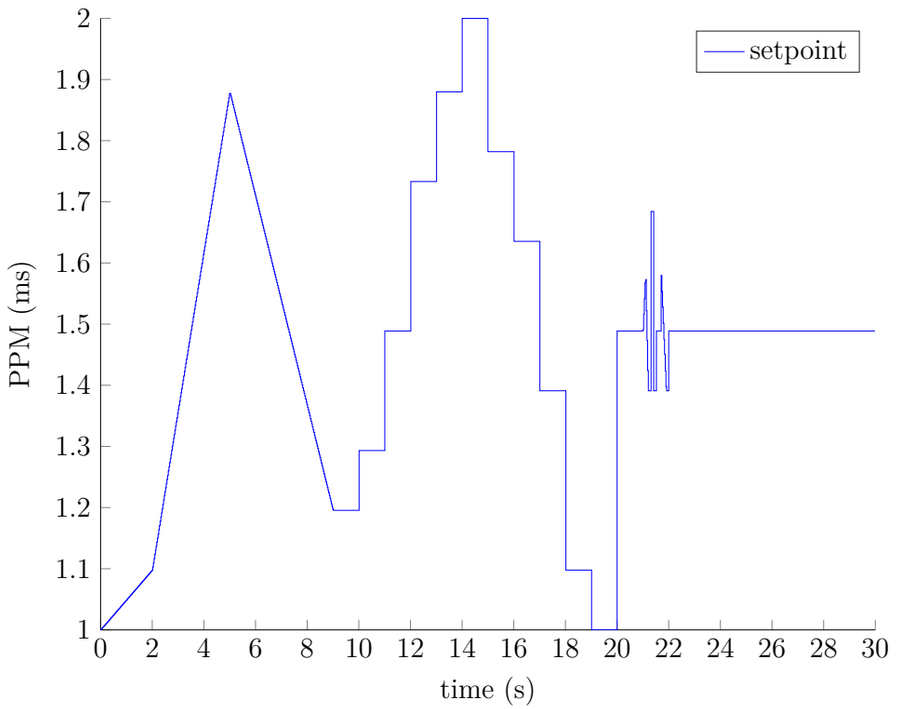


Figure 6.9 – Profile input for the tuning of the motor control loop.

drop was manually applied to the power supply, it is not synchronized between all the experiments.

- Fig. 6.11b shows the response to a step input, at $t = 11$s the PPM signal input of the ESC goes from 1.29 ms (29 % full speed) to 1.49 ms (49 % full speed). The open-loop response has not been plotted because it does not converge to the same value.

The parameters which have been found to give the best performances in terms of both disturbance rejection and step response are Kp = 3, and Ki = 3. A high Ki tends to provide faster disturbance rejection but it leads to an important overshoot of the step response if the Kp is not high enough.

Firmware

The firmware is based on Multiwii card, however the code has been written to run on numerous platforms and flight systems. In order to edit the code, first it is necessary to specify which type of multi-rotor is used. For this, two software packages are needed:

- **Arduino**: the development environment which allows to edit and upload the code;

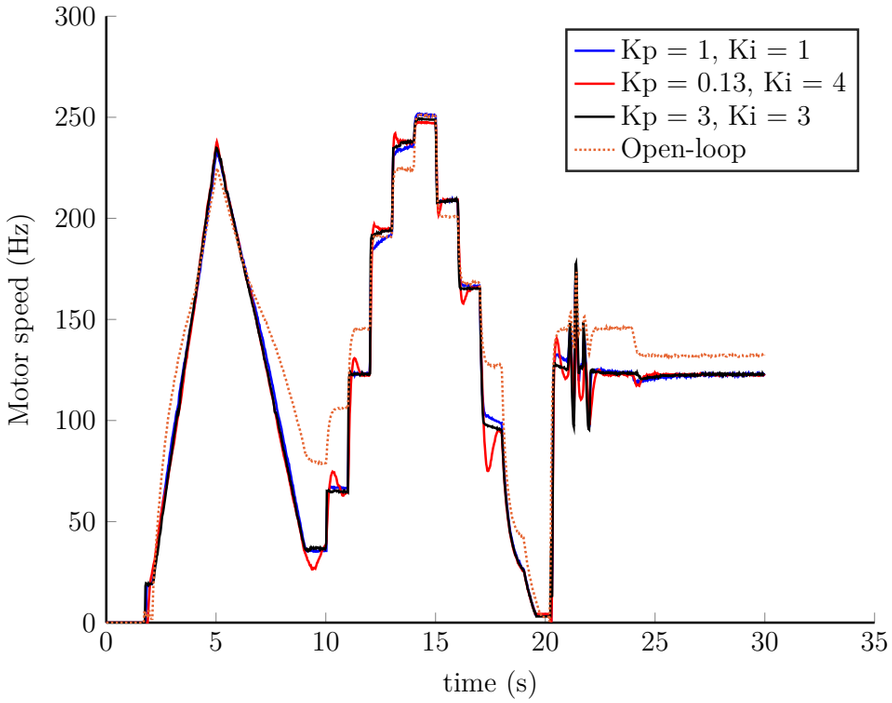
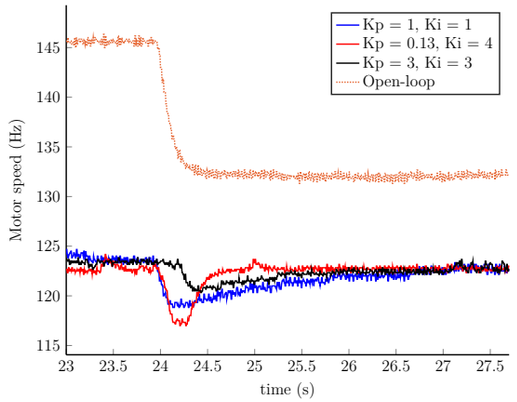
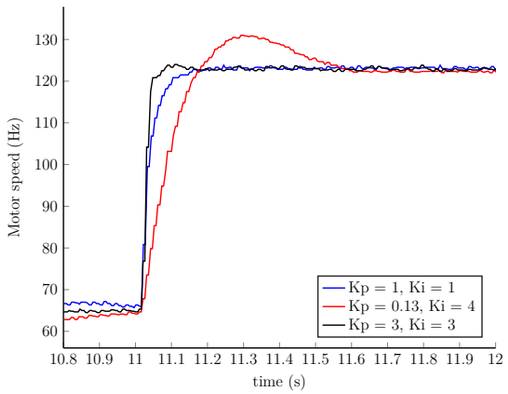


Figure 6.10 – Measured speed for different tuning parameters.



(a) Disturbance rejection. A 1 Volt drop is applied to the power supply, please note that the drop is not applied at the same time between the different experiments.



(b) Step response. At $t = 11$s the PPM signal input of the ESC goes from 1.29 ms (29 % full speed) to 1.49 ms (49 % full speed).

Figure 6.11 – Tuning of the ESC's gain.

- **Multiwii**: it includes both the open source code and the graphic user interface (GUI), necessary for the configuration of the board parameters

Multiwii GUI

To change the card's flight parameters, the MultiWii graphical interface is used, which is activated running the file MULTIWIIConf.exe. Once the board is connected to a COM port of the computer, parameters of the different sensors in the board (accelerometers, gyros, magnetometer, altimeter) can be changed. In general, the main window of the Multiwii GUI, can be seen in Fig. 6.12

Figure 6.12 – Multiwii GUI.

6.3 Experimental results

In this section some practical results are presented in order to test the vision system and the control laws. First, a quasi-virtual scenario is developed to keep up safety of quadrotor system and the pilot during tests, comparing data provided by the vision system with respect to the acquired data given by the Vicon system, and to perform the vision system employing a virtual image from a virtual environment, a more detailed scheme is shown in Fig. 6.13.

In this experiment, the flight is also made into the space of the Vicon system. The data estimated from vision system are also compared with the acquired data by the Vicon system, see Fig. 6.14. Finally, an experiment in a real corridor is accomplished, but head-direction was only controlled using data coming from our vision algorithm.

6.3.1 Vicon tracker system, ground station and quadrotor

Vicon tracker system estimates the position and orientation of the system. These data are sent to the ground station which uses XPC-target for computing the position control

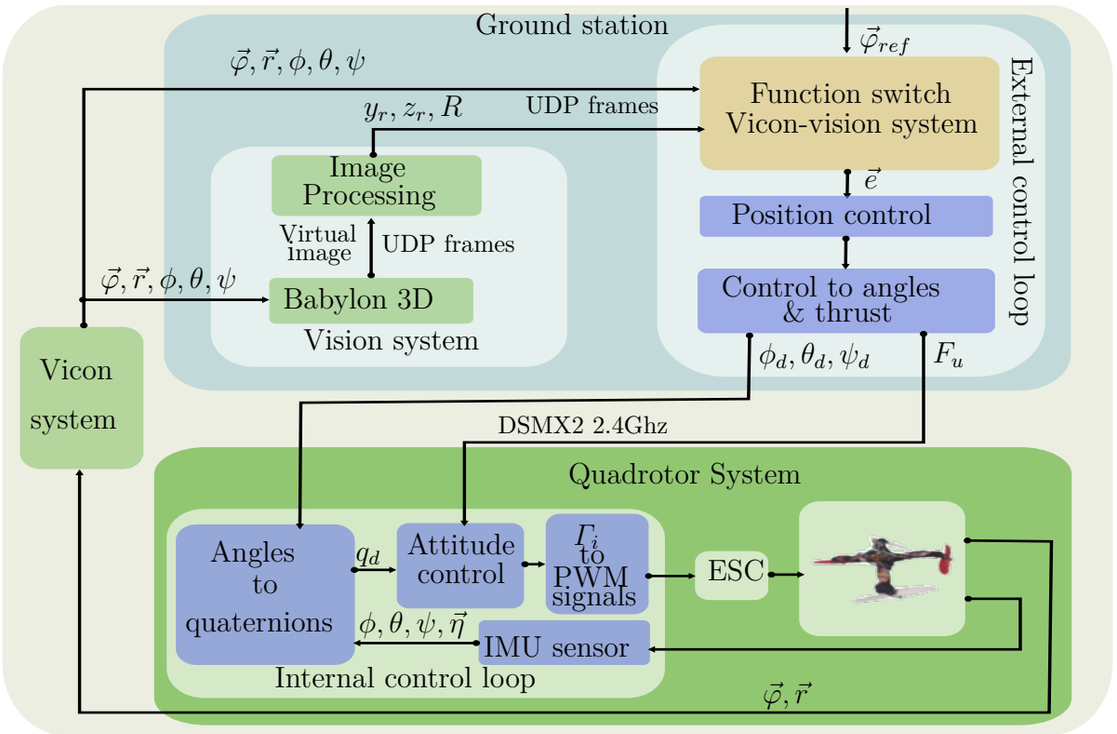


Figure 6.13 – General scheme of the vision system switching with the Vicon system using images from virtual world.

algorithm. Generated signals are sent to the quadrotor via radio signal. Even though all quadrotor states are available, meaning that the attitude control could be also computed on the ground station, the attitude controller is computed on-board of the quadrotor. The aerial prototype used in this experiment was described in section 6.2.2.

6.3.2 Quasi-virtual scenario

BabylonJs 3D Simulator is a complete JavaScript framework for building 3D games with HTML5, WebGL and Web Audio. It is executed in a specific computer to combine the simulator with the real world. Here a real quadrotor flight is run with image coming of a virtual world, called quasi-virtual flight. In this test quadrotor position vector obtained from the Vicon System (real word) is used into a *s*-function, which is made using MATLAB, for moving the internal camera of the simulator (virtual world), covering the vehicle and involving it with the simulator in a quasi-virtual scenario. This computer is also capable to project in a screen the virtual environment (in our case a corridor) moving in concordance with the real object. The acquisition image is at 200 fps.

Once the vehicle and the virtual scenario are connected, the vision algorithm can be executed, lateral states , altitude states, and head-direction are estimated using only

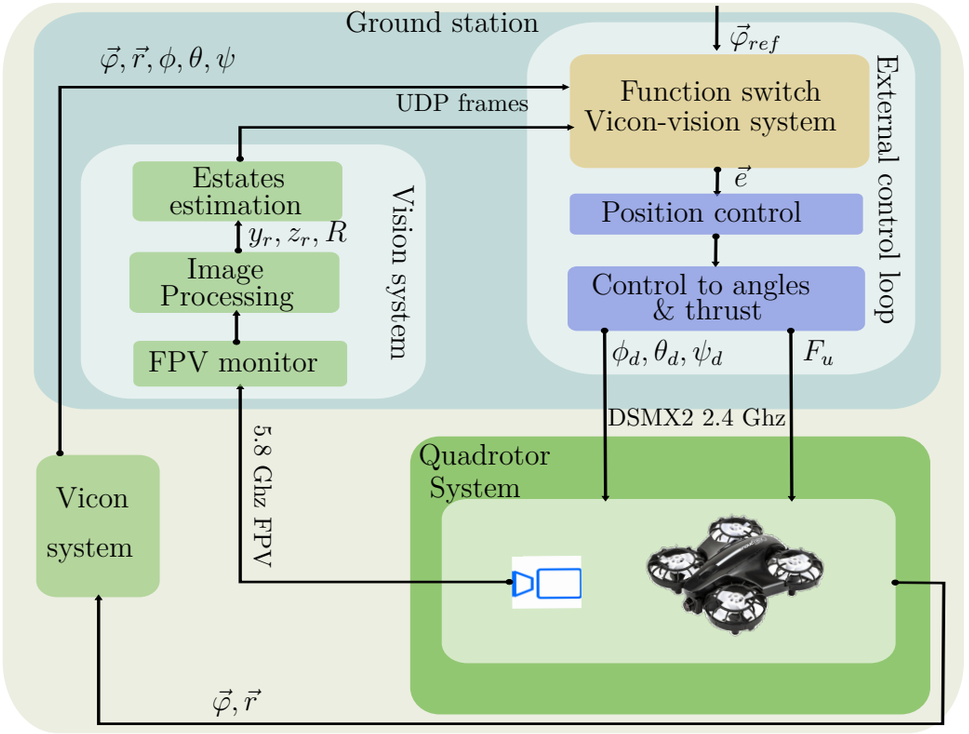


Figure 6.14 – General scheme of the vision system switching with the Vicon system using images from real world.

the virtual images emulating a flight into a corridor, see Fig. 6.15. For maintaining the aerial vehicle in the middle of the corridor, the control algorithms are also computed for stabilizing its orientation and moving the vehicle into the corridor.

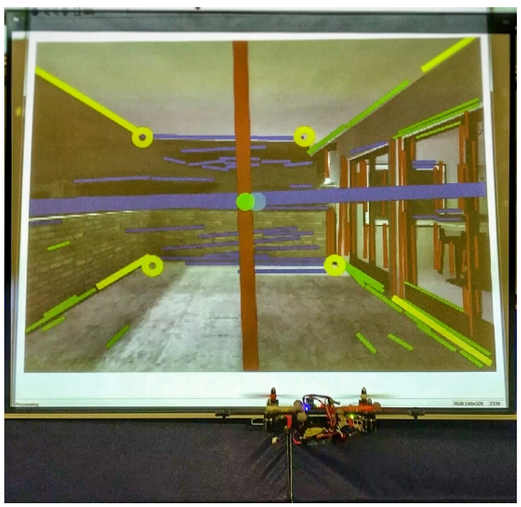


Figure 6.15 – A quasi-virtual flight with autonomous navigation into a corridor.

The advantage of this scenario is that it can be changed by other 3D scenarios defined by the user.

Therefore, experimental tests were carried out for validating the vision algorithm using collinearity between lines. In order to obtain the best result with the vision algorithm, it needs to be executed when the vehicle is flying. To simplify, the simulation and the experimental tests, the x coordinate and the pitch movement are fixed for limitations in the workspace. Then, the quadrotor will move as a PVTOL aircraft evolving in the $z - y$ plane with a roll angle, and the ψ angle will take the vanishing point direction which has a value around to zero. This means that the states x and θ are stabilized with the controller using the measurements coming from the Vicon system, or controlled by an external pilot. The vehicle is in the middle of the virtual corridor when the position provided by the Vicon system is at $x = 0$, $y = 0$ and $z = 0.9$ all in meters.

The test procedure is the following: The quadrotor is placed, using the data acquired by the Vicon measurements (quasi-virtual system), in different positions in the y and z coordinates at different times as shown in Table 6.3 (quasi-virtual tests). Later, these measurements (only for y and z) are switched with those (y_r, z_r) estimated by the vision algorithm. As the control goal is to keep the vehicle in the middle of the corridor, then the control laws are computed using only the (y_r, z_r) measurements for carrying the quadrotor at the center of the corridor.

Table 6.3 – Position to quasi-virtual tests

Initial position	1	2	3	4	5	6	7	8
Time (s)	0	12	75	85	110	185	163	170
y	0.5	0	-0.6	0	0.6	0	-0.6	0.0
z	0.5	0.9	1.3	0.9	1.3	0.9	0.6	0.9

6.3.3 Quasi-virtual test

This system is shown in Fig. 6.16 where the virtual image is projected on the wall to have a more visual experiment; though, this mean that images are taken using a real camera, but image processing keeps virtually.

Similarly to simulation test, Fig. 6.17 shows the evolution of the different positions. Positions $1, 3, 5$ and 7 from the Table 6.3 represent the flag at 0, and the values obtained by the Vicon system are used to stabilize the system. Furthermore, positions $2, 4, 6$ and 8 represent the flag at 1 and the values estimated by the vision system are used for centering the system to the origin of the vision system.

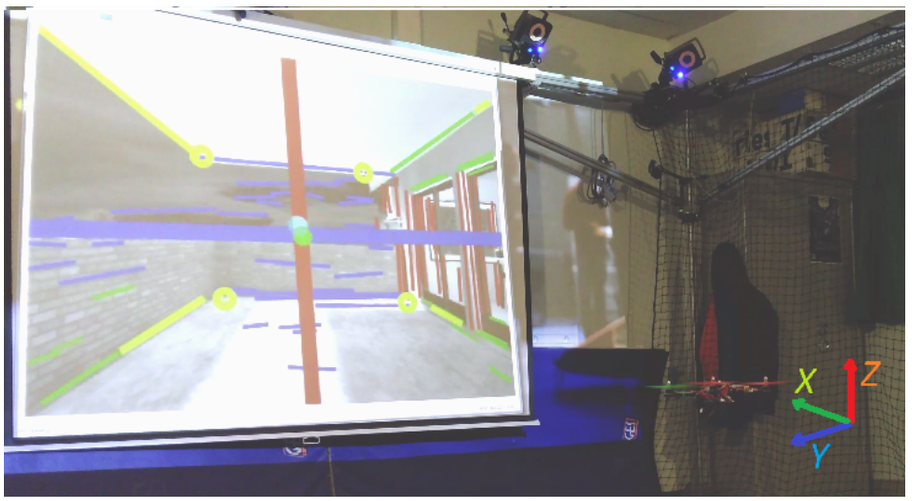


Figure 6.16 – Vision system and vehicle running with virtual image.

The real y position is shown in Fig. 6.17(c). Although the variation is similar to values coming from the estimated vision system, see Fig. 6.17(b), the $2, 4, 6$ and 8, the positions represent the origin from the vision system being similar in both figures. Furthermore, the real z position is represented in Fig. 6.17(c). However the variation from the Fig. 6.17(b) is different to the values from the estimated vision system due to the change of the origin (vision system). This information serves as the error input into the proposed controller. Fig. 6.17(d) depicts the evolution in terms of error. There is a variation at 15s, 58s, 108s, 135s, 160s, and 169s in y and z axes in all the figures, but in Fig. 6.17(d) it can be observed the instant when the y and z data are switched. The position $1, 3, 5$ and 7 represent the position in the $y - z$ plane (1-right down, 3-left up, 5-right up and 7-left down).

In others words, the test is switched between real data and estimated data during all the experiment . Fig. 6.17 (e) represents this commutation. Where "0" means that data is taken from Vicon system, and "1" denotes that data is acquired from Vision system. Then, the experiment starts using Vision data (flag= "0" in Fig. 6.17 (e)). The quadrotos is stabilized in position $x = 0$, $y = 0.5$ and $z = 0.5$. It has to be mentioned that x-axis is always controlled using data from Vicon system. Fig. 6.17 (c) depicts real position from real system. Fig. 6.17 (b) represents relative position from Vision system. Data from both system, Vicon and Vision systems, are quite different because they have a different origin. In this case, the origin of Vision system is placed in the center of the corridor or position (0,0,1) meters (Fig. 6.17 (c)) from data of Vicon system. The error is mostly zero as it can be seen in Fig. 6.17 (d). Then, in time 18s approximately, Vision data is commuted (flag of Fig. 6.17 (e) is now "1"). Fig. 6.17 (d) shows an increase in the error because data reference was changed, however the stabilization is quite fast, and the system converge to zero again. This can be seen in Fig. 6.17 (b) where Vision data is stabilized to its origin.

Fig. 6.17 (c) represents the cross product which is the collinearity between principal lines. When position control takes Vicon data as input reference, the collinearity takes a valor different to zero, but if the reference is changed to our Vision system, the system will converge to the collinearity of the principal lines.

This test is shown in a different way in Fig. 6.18, where the acquired image keeps virtual which means that this time the images are not projected on the wall.

A video of this experiment can be see in TELLEZ [2018].

6.4 Test: Using real image

This test presents some results using real images. These images are acquired using an on-board camera of the system presented in section 6.2.3. The experiment was carried out in a structured environment (Moca room). Fig. 6.19 depicts a sequence of images during the drone flight. These images show that the environment has some similitude with respect to a corridor. Position and head direction are estimated using our proposed algorithm. Fig. 6.20 shows the behavior of the vision system using the same sequence of images. The size of images is 240x360 pixels. It can be also observed that there is a low lighting, and some images are really noisy. Nevertheless, our algorithm is capable to extract the right lines to estimate the rotation matrix and relative position using the geometry environment.

Using the rotation matrix, the head direction (ψ) can be controlled to keep the quadrotor pointed to the finite vanishing point, which is near to the image center. The test is made using position states given by Vicon system, and at time 20s the pose estimation algorithm is used to control the position of the system. Fig. 6.21 shows that y_r and z_r data are switched at time 20s to close the position control loop in y_n and z_n axes. x_n-axis is controlled using Vicon data.

Fig. 6.21a shows real position which is acquired from Vicon system. z-axis was set to 1m. y-axis was put to 0.5m until the time 90s, and then it was displaced to one side at time 92s. At time 103s position of y-axis was returned to 0.5m. Position was newly moved to the same side at time 133s. Finally, y-axis was posed to the initial position at time 142s.

Fig. 6.21b depicts relative position estimated by our proposed vision system. In this image, z-axis is quite different, because the origin of vision system is not the same with respect to the Vicon system. In the particular case of a corridor, the origin of the vision system in real world, using our algorithm to estimate the relative position, is located in the center of the corridor. Then, the information of z-axis acquired with respect to the origin of vision system is nearly to zero. y-axis is also posed in the origin of vision system,

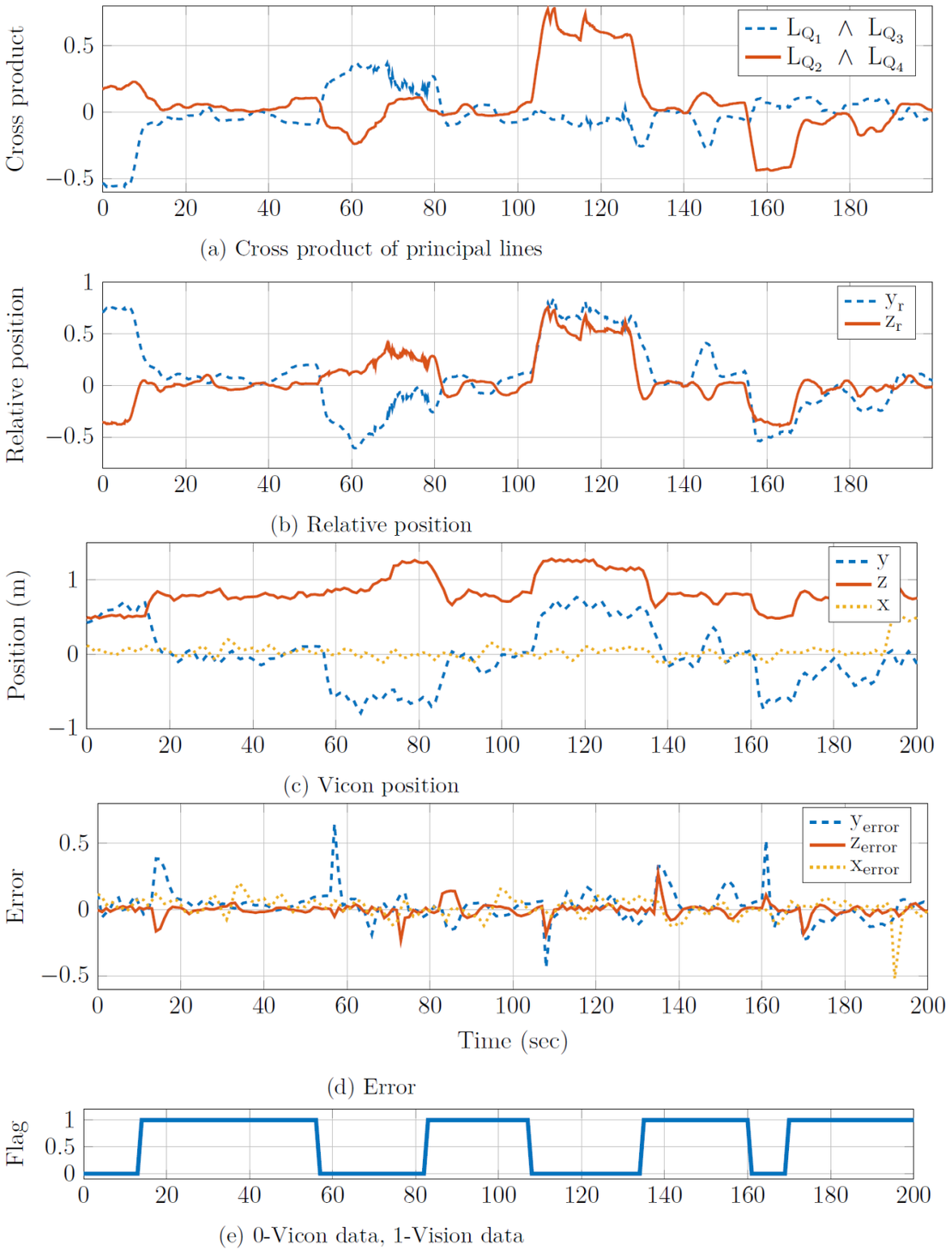


Figure 6.17 – Stabilization using vehicle-Vicon system and Babylon 3D.

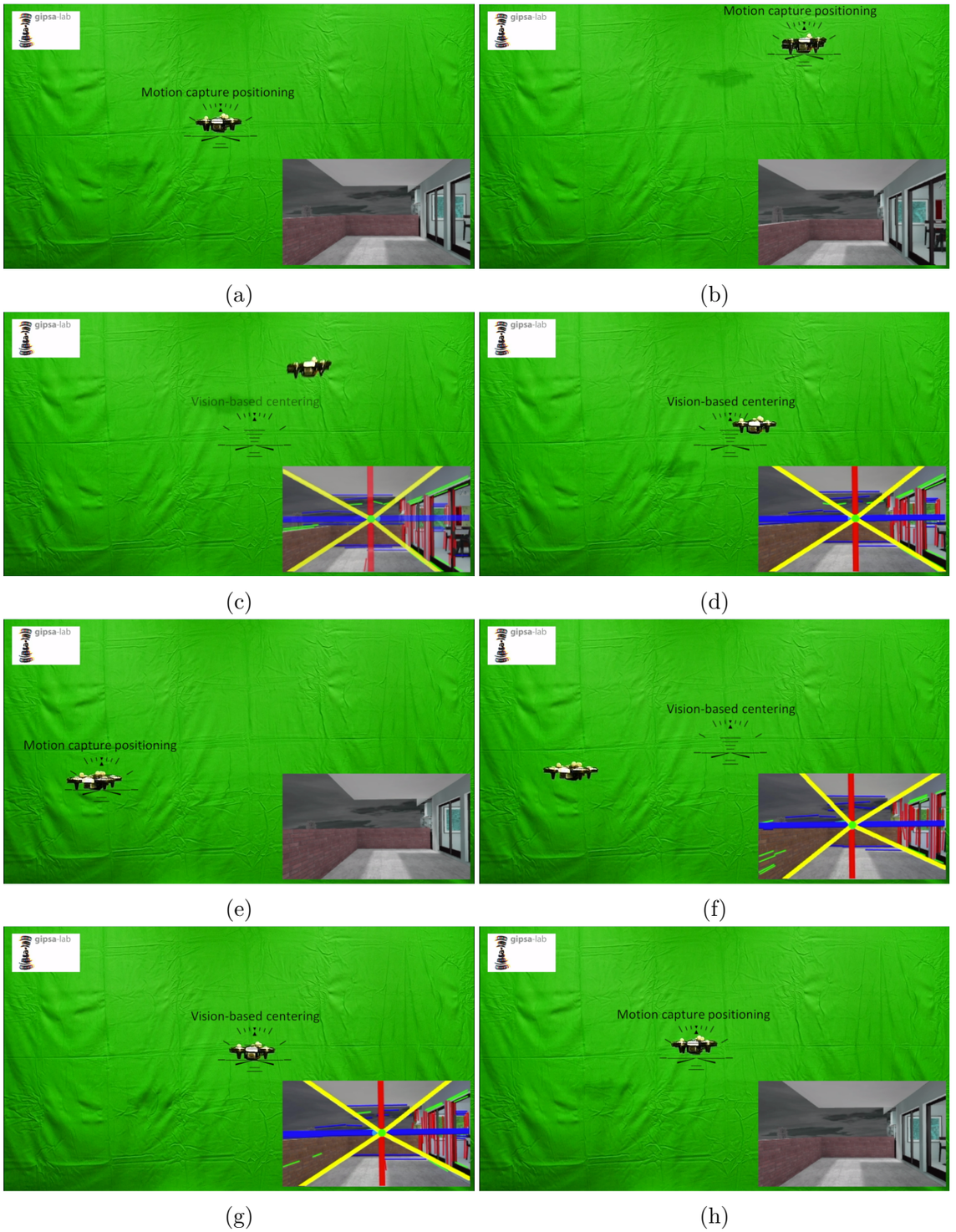


Figure 6.18 – Quasi-virtual system. In (a), (b), (e), and (h) it is used motion capture to closed-loop control. Vision-based centering is applied in (c), (d), (f), and (g).

but there are some variations because of the displacement in its real position.

Finally, Fig. 6.21c represents the comparison of real and relative position. The information from relative position was added an offset to match with real position in order to make more visual theirs equivalences. Even though, information from relative position is a little noisy with respect to the real position, the data is very similar. It should be noticed that vision system is not using any filter to improve its performance.

Figure 6.19 – Sequences of images from structure of salle moca.

6.5 Test into a corridor

Fig. 6.23 shows a sequence of images when the system is flying inside a corridor. The test is performed using head direction autonomously. If the on-board camera is always pointing to the end of corridor, finite and infinite vanishing points can be extracted to estimate the rotation matrix. Thus head direction (ψ) can be computed, and it can be used to navigate along the corridor.

Fig. 6.22 shows the same sequence from Fig. 6.23, but now running the vision system. Even though there are several movements during the test flight, head direction always

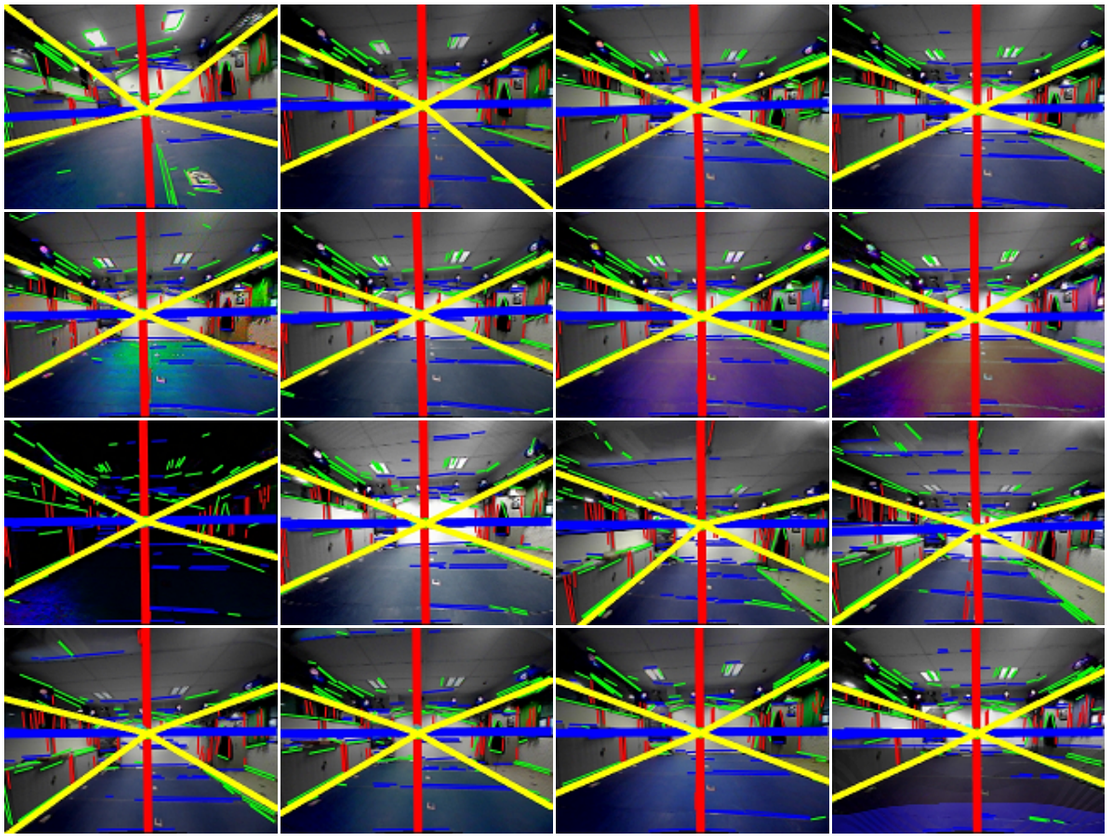
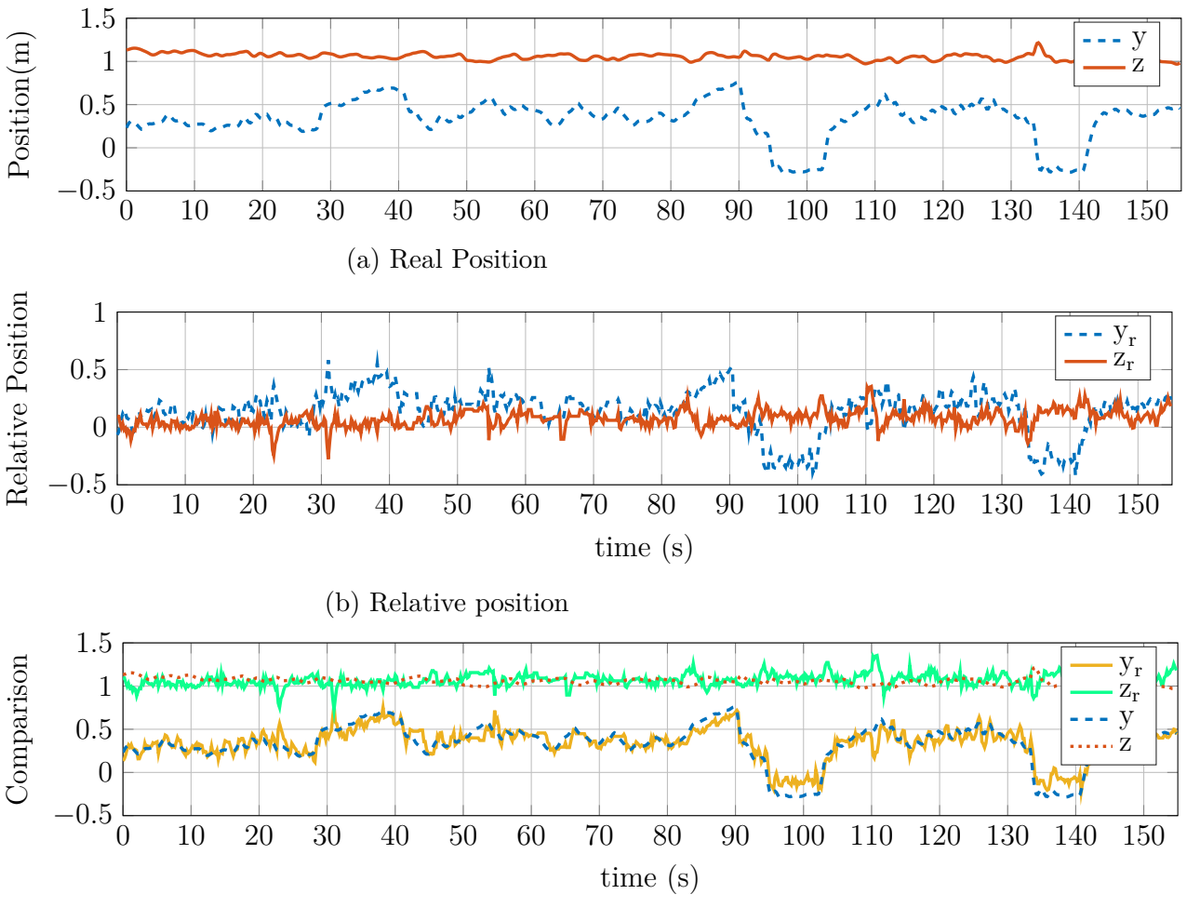


Figure 6.20 – Sequences of images from structure of salle moca, applying the proposed algorithm.

holds the right direction. Meaning ϕ is roughly equal to zero if the head direction is pointed directly to the end of a corridor, as it can be see in Fig. 6.24. Quadrotor movements are performed manually on axes y_n, z_n in plane image and even on depth x_n.

6.6 Summary

This chapter was focused on describing the platforms and our experiments, where some tests were carried out to validate theirs performance. Firstly, the platform which consists on a ground station and a quadrotor system was detailed. The elements that describe the ground station were presented. Moreover, it was mentioned that Moca system is used to extract the real position of the system, and it serves to compare the estimated position. Later, a proposed algorithm was running in a workstation to manage all the data and to compute the position control. Another algorithm running in a PC is employed to compute the proposed vision algorithm. Finally, some experiments using virtual and real images like inputs were performed to test our proposed algorithm.



(a) Real Position

(b) Relative position

(c) Real position (y, z) vs relative position (y_r, z_r)

Figure 6.21 – Comparison between Vision data and Vision data using real video

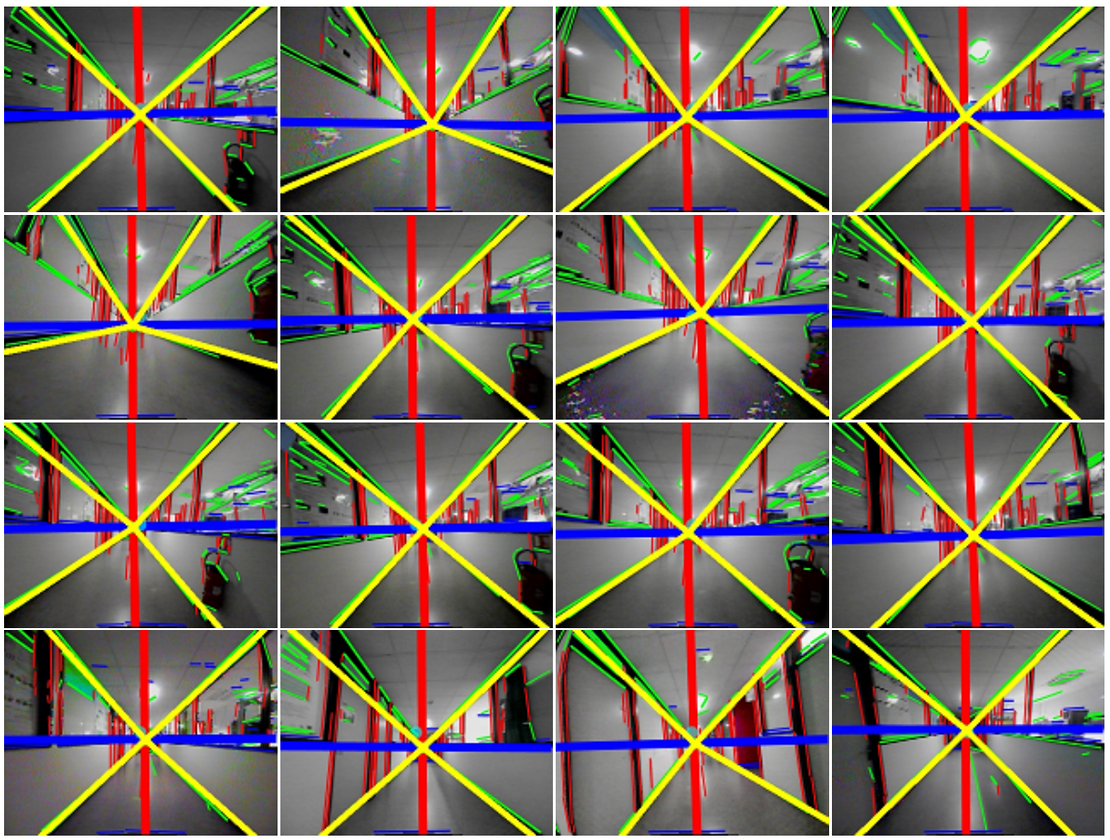


Figure 6.22 – Images frames after processing from real time video starting from upper left corner until lower right corner

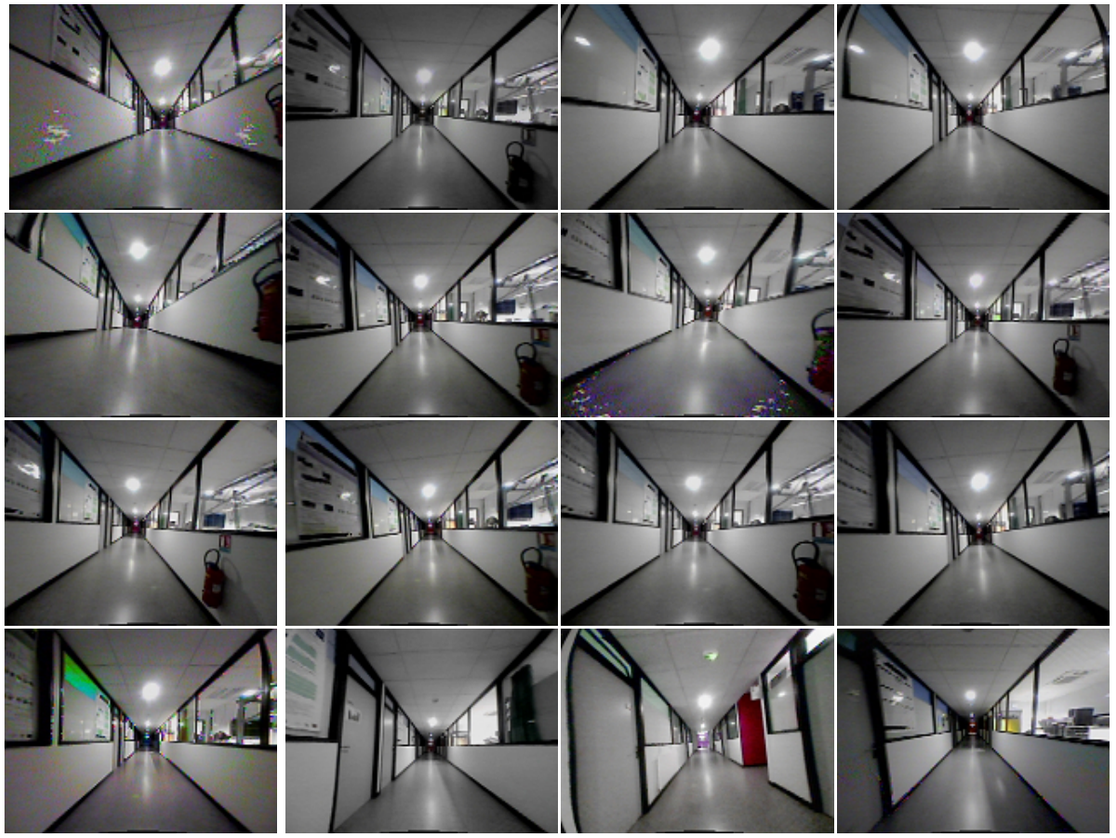


Figure 6.23 – Images frames from real time video starting from upper left corner until lower right corner

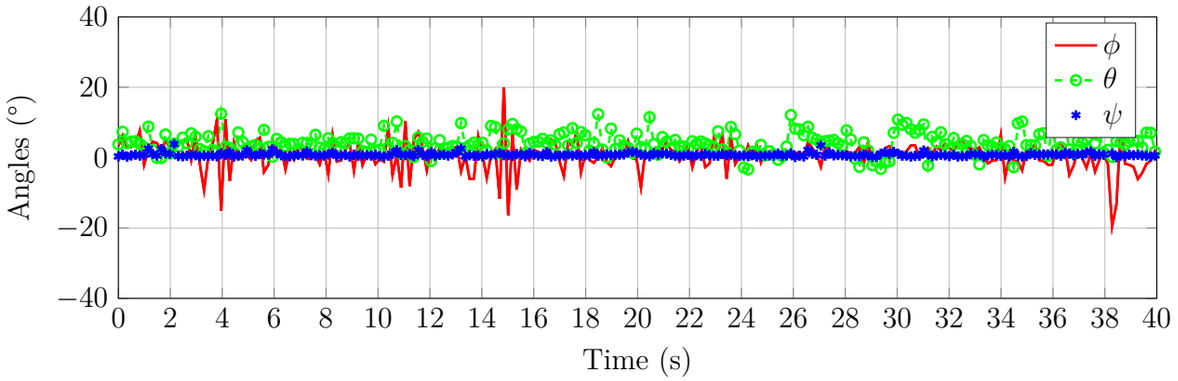


Figure 6.24 – Quadrotor angles estimated by proposed algorithm using images taken by embedded camera, roll ϕ, pitch θ, yaw ψ. Yaw angle is only near to zero due to heading control proposed in this work.

Chapter 7
Conclusions and Future work

7.1 Conclusions

This thesis was thought as an alternative to perform an indoor navigation of a quadrotor system using an embedded camera. The proposed vision algorithm was also designed in such a way that the computational cost is minimum to optimize our proposed embedded system. Thus, a vision algorithm, which uses perspective geometry from real time video sequence, was proposed to solve partially the problem of localization, and then, try to navigate over a corridor autonomously. For that, we proposed a quadrotor system that evolves into a structured environment to use parallelism of hand made structures. From a geometrical point of view all parallel lines in 3D world converge into a geometrical point named vanishing point in camera image 2D plane. To develop a quadrotor navigating in real world some research subject were studied and established:

In vision:

- Perspective projection: The simple form to interpret the orthographic projection is using the perspective projection. This transformation represents a method to render a point in 3D dimension to a pane of 2D. Moreover, this concept also serves to explain how the human eye sees a scene, whose objects in the 3D seem to be small in the distance. That means, while orthographic projection ignores this effect to allow accurate measurements, perspective projection shows distant objects as smaller to provide additional realism. However, distance and shape are not conserved, but straight lines are kept. Parallel lines in the image plane always converge into a point called vanishing point, which is used like a reference and in our case for 3D reconstruction. Section 2.1.4 from Chapter 2 is focused on this concept.

- Camera calibration: Extrinsic and intrinsic parameters can be obtained using an

algorithm for camera calibration. These parameters are implicit in the camera matrix. Intrinsic parameters give the relation between a 3D point in the real world to a 2D point in the image plane, and extrinsic parameters yield a matrix where rotation and translation are implicit. This matrix gives the camera motion with respect to the world. These data can be normally extracted with an algorithm for camera calibration using vanishing points. Intrinsic parameters can be only obtained under certain scenarios, but extrinsic parameters can be extracted. Taking into account that the scenario is into a corridor, and considering the perspective projection of the environment, a partial camera calibration can be made in order to obtain the motion of the camera with respect to the real world. See Chapter 2.

- 3D reconstruction: The obtained information (points, lines, color, texture) from an image gives certain information to extract in a succinct way the 3D information of the world. Usually, a common structured environment (as a corridor) is plenty of lines whose directions coincide with the shape of environment. These directions converge to a point called vanishing point, and they also meet with the orthogonal direction of the 3D world. Using some strategies of camera calibration, the orthogonal directions are useful to rebuild in a simple way the 3D world. The proposed algorithm was showed in Chapter 4.

- Lines segmentation, see Section 4.2.1: Once the image is acquired by the onboard camera on quadrotor system, the next stage is to extract all the useful information to interpret the 3D world using data from an image plane. The environment can be described by a set of lines which can extracted using detector lines algorithm. Hough transform is the common tool to identify lines by a voting procedure. Even, this approach is useful to find other shapes in the image plane, sometimes it has a high computational cost. Furthermore, an edge detector (Canny, Sobel, etc.) has to be used before it. Nowadays, in the literature some strategies have been developed by the community research, for instance, EDlines is an algorithm developed to work in real time. The information yielded by EDlines is a set of points which describes the lines extracted in the environment. The information extracted by some detector lines can be used to describe the real world through lines.

- Lines clustering: The obtained data for the line detector is a set of lines without any order. In this case in particular, lines clustering was focused to find the parallels lines converging in a vanishing point. In this work due to restriction of our goals, only a simple classification according to the slope of each lines is proposed. The result is a set of vertical, horizontal and diagonal lines which give us the data to obtain the vanishing points. This information was used to extract the intrinsic and

extrinsic parameters of a camera. In addition, this classification was also applied to compute the edges lines used to extract certain position in y and z axes, see Section 4.2.2.

In control:

- Data fusion from IMU and from vision data: To have knowledge about all the states of the system using a single modality is a hard task. In the particular case of a quadrotor system, which needs orientation and position states to perform a control in both. Thus, even there are some works where it is used only an IMU system to estimate attitude and position states, we propose to use two different modalities for each. Therefore, IMU was used to obtain orientation, and a vision system was employed to acquire a partial position and head-direction of the quadrotor system. It should be mentioned that vision system is capable to yield the orientation of the system, but these data is not used due to its processing speed. For safety reasons, IMU data is used to close the loop of the orientation control law. Nevertheless, head-direction obtained by vision system is used to ensure the right direction of the quadrotor system w.r.t. the world. Keeping the desired direction, vision system is able to obtain certain position in y, z axes with respect to the environment (corridor). This approach can be seen, in general way, like a multimodal system since it has has two modalities to complete the vector state. In Chapter 5 were presented some schemes using both IMU and Vision data, and in Chapter 6 were analyzed some results using these data.

- Closed loop control: Two control laws have as been implemented in order to have an autonomous flight trough a corridor. A quadrotor system can be considered as cascade control system. That means, we can develop an attitude and position control separately; however it has to be taken into account that attitude control does not depend on position control, but position control relies on attitude control. Furthermore, as the frequency of the vision algorithm is relatively slow (20Hz), the proposed control laws takes into account this drawback. For attitude control, IMU data is used to close the feedback control. For position control, data vision was applied to pose the system in middle of a corridor. Both control laws were using bounded control strategies, see Chapter 5.

- Open loop control: A quadrotor system can be partially controlled as a non-feedback configuration. Since these systems are considered like a cascade control system, position control can be commanded as open loop control. However, attitude control can not be controlled using the same strategy because of its dynamic model. In this

work, vision system is only able to estimate position in two axes (y, z). Thus, a closed loop control system is used in (y, z), and open-loop system could be used for x-axis. It should be mentioned that x-axis can be obtained using other source or sensor (MOCA system) to perform a full feedback control. x-axis is handled by a pilot for a non-feedback control. At the end of Chapter 6 was presented an experiment where head direction was only controlled taking as reference vision data.

- Data estimation: The information obtained by vision system (y_r, z_r) can be used as inputs in the feedback control for position in y_n, z_n axes. However, velocities in both axes are missing since our vision system is not able to estimate directly this information. Thus, to start a first order filter was applied to the data obtained by vision system, but noisy from estimated signals affect the final result. Therefore, a Kalman filter, which is used to solve common problems for navigation, guidance and control, can be used to estimate the velocity in (y, z) axes.

Contributions:

- Autonomous navigation into a corridor placed in the center of the corridor. The proposed platform can keep its position in (y, z) axes w.r.t. corridor. Taking into account that origin is considered to be in the center of a corridor according to camera perspective. The position control is able to stabilize to the center of the corridor.

- Stability of quadrotor using data fusion visual and inertial data. Even though attitude control uses as inputs IMU Data, head direction is controlled by information estimated by vision system. Furthermore, (y_r, z_r) estimated by vision system are applied to feedback the position control in (y_n, z_n) with respect to the corridor, and x_n is left as non-feedback control.

- Merging real platform evolving into virtual world: Quasi-virtual system. The platform developed in this thesis work was made for several stages. Firstly, a full simulation of the system, which comprehend the model system, attitude and position control, and a virtual environment. The 3D world was created in order to make more visible the behavior of the system and to acquire the frames images created by a 3D engine . These images are used to obtain the partial position of the quadrotor system. Then, a combination of a real system and virtual system was implemented. In this case, it was used a real quadrotor system, but virtual images were still obtained from virtual environment. Moreover, vicon system was also used to compare data from vision system with Vicon system. Finally, virtual images were replaced using real images acquired by camera embedded on quadrotor system.

- Quadrotor development taking into account, dynamical and mechanical system. In the beginning of this thesis several structures were designed which were made with a 3D printer from GIPSA-LAB. These structures were drawn according to its size and payload capacity.
- Embedded system implementation using low images. CRIUS flight controller was used as embedded system. Attitude control was only implemented in this card. However, the system has also embedded a small camera which send the image video via wireless to the ground station. The embedded system is also configured with inputs and outputs to interact with the world. For instance, the protocol DSM-2 is used to send and receive signals of reference to control the quadrotor.

Publications

Some publications were produced, namely,

- Conference paper
  - Velocity control of mini-UAV using a helmet system Tellez-Guzman et al. [2015]
  - Nonlinear control of a nano-hexacopter carrying a manipulator arm Alvarez-Muñoz et al. [2015]
  - Integral Backstepping Control for Trajectory Tracking of a Hybrid Vehicle Colmenares-Vázquez et al. [2015a]
  - Position Control of a Quadrotor under External Constant Disturbance Colmenares-Vázquez et al. [2015b]
- Journal
  - Rotorcraft with a 3DOF Rigid Manipulator: Quaternion-based Modeling and Real-time Control Tolerant to Multi-body Couplings Alvarez-Munoz et al. [2018]

and "Quasi-virtual UAV autonomous navigation into a corridor:A vision collinearity approach" is in reviewing to submit to a journal publication.

7.2 Future Work

Even the partial navigation of the quadrotor system could be carried out successfully using only the visual information of a camera and a vision algorithm based on perspective projection, there are some problems to solve, which are mentioned below:

- Firstly, our vision system is capable to estimate y, z position and head direction, but linear velocities are missing. To solve this problem, we are working in a filter Kalman to filter data and to estimate linear velocities.

- Secondly, there are some problems of delays due to communication and computation of the vision system. Thus, we are designing a system predictor (Smith predictor). Particularly, we are working with a merge between Smith predictor and Kalman filter.

- Finally, a complete embedded system will be used to implement all the vision system.

Appendix A
Experimental platforms

A.1 Flexbot

Flexbot comes from a crowd-funding project dedicated to offer open source hardware multi-copters controllable with a smartphone. Fig. A.1 shows the Flexbot hexa-rotor and quad-rotor platforms.

Figure A.1 – Flexbot platforms, hexa-rotor and qua-drotor.

Hardware

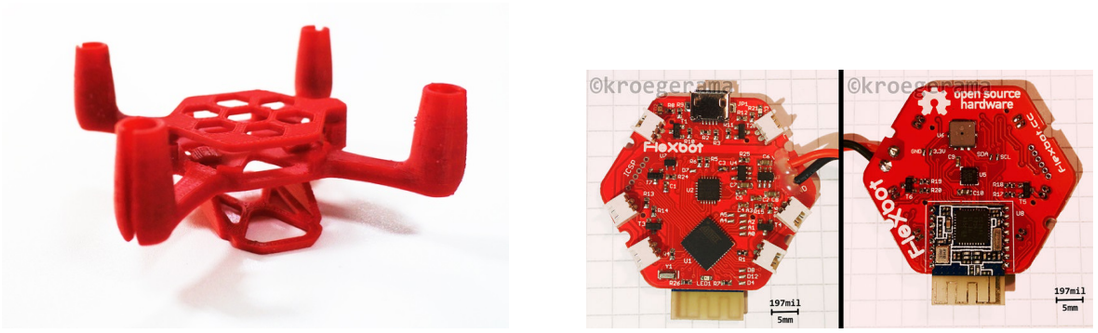
Some of the main elements of these platforms are presented below. We present a brief introduction of the frame, flight controller board and communication protocol.

Frame

The frame is 3D printed and the CAD model is open source. The frame of the quad-copter is shown on Fig. A.2a

Flight controller board

The flight controller board allows the implementation of the attitude control law, since the on-board software is equally open source. The board features an ATMega32u4 processor, an IMU sensor (MPU6050), a magnetometer (HMCL5883L) and a barometer (BMP085). The motors are directly connected to the card as some transistors are mounted to run the DC motors. The board can be programmed with arduino, and the default firmware is based on MultiWii. A blue-tooth low energy module is also mounted and communicate through a serial bus with the processor. Fig. A.2b shows the flight controller board.



(a) 3D printed frame. (b) Flight controller board.

Figure A.2 – 3D printed frame of the Flexbot quad-rotor and the flight controller board.

Communication

An application for smart-phone is provided to control the multi-copter with blue-tooth protocol. The data received by the flight controller board is organized with *MultiWii Serial Protocol* (MSP). In order to communicate between the ground station and the platform, a specific board had to be developed. It consists on an arduino MEGA, an Ethernet shield and a blue-tooth shield. The arduino board is connected to the same local network as the ground station which packs the commands to fulfill the MSP and sends it through UDP to the IP adress of the arduino board at a specific port. The arduino board is running an UDP server which listens to the communication, pairs the blue-tooth device and relays the data received by UDP to the Flexbot's blue-tooth module.

This solution has been found to be reliable, but was only possible because the blue-tooth module of the Flexbot and the arduino shield were both the same. However, Flexbot changed the bluetooth module after the first batch, the bluetooth shield was then not able to be paired with the flight controller board. To overcome this issue, a Raspberry Pi and a bluetooth dongle have been used. The communication protocol had to be reverse engined, a python script has been developed to run an UDP server, and to send the data through bluetooth protocol.

A.2 Other experimental platforms at GIPSA

Currently there are three other projects that gave way to the usage of other experimental platforms.

The main objective of the first project was the development of a control law for a hybrid vehicle, which allows to the system fly and move on the floor automatically. It was developed by Josue Colmenares. Two prototypes were used during the development of this project. The first one consisted on the FLEXBOT nano-hexacopter, see section A.1. The second one consisted on a hybrid vehicle called the "Wagon" hybrid vehicle, see Fig. A.3. As in our case, the frame was designed and 3D printed in GIPSA-lab.

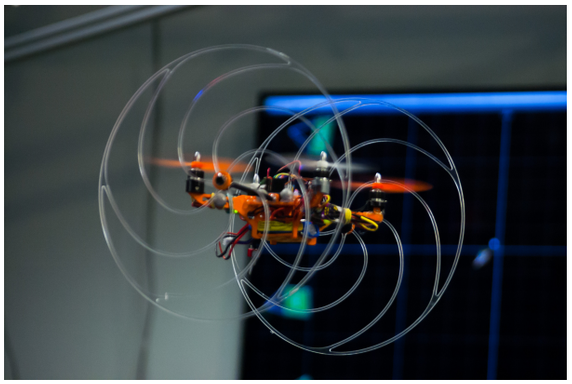


Figure A.3 – Wagon hybrid vehicle in flight

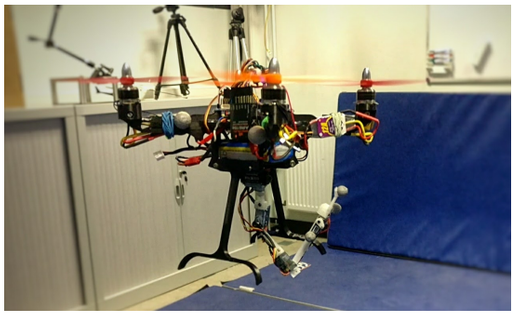
The objective of the second project was the design of an event-based control for micro biometric robots. The development of this work was in charge of Bruno Boisseau. Many experimental prototypes were tested during the development of the project, including two nano-quadrotors called "Inductrix" and "Nano-QX" both from BLADE, see Fig. A.4. BLADE is a company subsidiary of Horizon Hobby, dedicated to the design and construction of mini UAVs.

The third project consists on a home-made quad-copter and a 3-DOF arm manipulator, named "Aerial Carrying robot" (Aeca). The development of this project is in charge of Jonatan Uziel Alvarez. Two different frames were designed and 3D printed. Since these designs have the actuators at different positions with respect to the quad-rotor center of mass, the objective of their construction was to test and validate a better flight performance with the manipulator arm. Both systems are shown in Fig. A.5a and Fig. A.5b.

(a)

(b)

Figure A.4 – (a)Inductrix nano-quadcopter and (b)Nano-QX nano-quadrotor in flight



(a) AeCa robot version 1.0 (Top located actuators).

(b) AeCa robot version 2.0 (Medium height located actuators).

Figure A.5 – 3D printed frames of the final prototypes.

Appendix B
Pre-experimental tests

B.1 Velocity control of mini-quadrotor using a helmet system

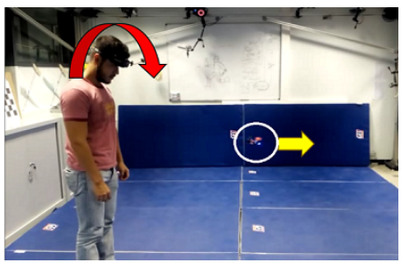
The usage of a helmet to command a mini-quadrotor, is a telepresence system that connects the operator to the vehicle. This test proposes a system which remotely allows the connection of a pilot's head motion and the 3D movements of a mini-quadrotor. Two velocity control algorithms have been tested in order to manipulate the system. Results demonstrate that these movements can be used as reference inputs of the controller of the mini-quadrotor.

B.2 Heading control

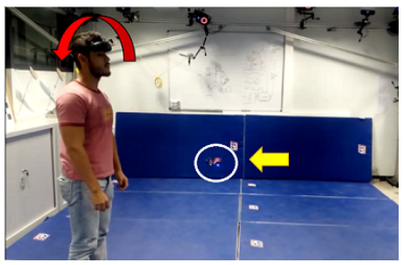
One of the characteristics of the head control with regard to quad-rotor control is that the operators can intuitively determine the position and the orientation of the mini-quadrotor. The operator wears a helmet and he can tilt his head to obtain the references of control. With these head motions, the operator can intuitively manipulate an quad-rotor. When the operator tilts his head in front or back (see Fig. B.1a and Fig. B.1b), right or left (see Fig. B.1c and Fig. B.1d) the UAV flies in the same direction. When the operator rotates his head, the UAV rotates in the same direction.

B.3 Control strategy

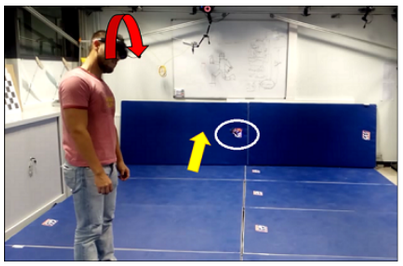
The mathematical model is detailed in section 3.5. The attitude control is described in section 5.3. Two position control laws are tested in this experiment. The first one is discussed in section 5.4. The second one is going to be explained below. This control law was used only during this experiment.



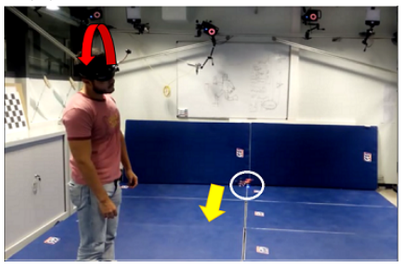
(a) Forward head movement



(b) Backward head movement



(c) Left head movement



(d) Right head movement

Figure B.1 – Teleoperation of quadrotor using head movements

B.3.1 Control Law for Trajectory Tracking

The approach used for the trajectory tracking is based in the backstepping technique. To ensure the convergence to desired trajectory using the backstepping design a integral control. The integral control helps to reduce the error of the tracking and add a factor to improve the robustness when parameters of the system are not well-known. In the next lines will describe the control strategy.

The thrust vector F and weight vector F_g are defined as follows:

$$F = \begin{bmatrix} 0 \\ 0 \\ f \end{bmatrix} \quad F_g = \begin{bmatrix} 0 \\ 0 \\ -mg \end{bmatrix} \tag{B.1}$$

The rotation matrix R is obtained from Euler angles in the order yaw-pitch-roll and it has the following expression:

$$R = \begin{bmatrix} c\psi c\theta & -s\psi c\phi + c\psi s\theta s\phi & s\psi s\phi + c\psi s\theta c\phi \\ s\psi c\theta & c\psi c\phi + s\psi s\theta s\phi & -c\psi s\phi + s\psi s\theta c\phi \\ -s\theta & c\theta s\phi & c\theta c\phi \end{bmatrix} \tag{B.2}$$

where $s = \sin(\cdot)$ and $c = \cos(\cdot)$. The yaw, pitch and roll angles are given by ψ, θ, ϕ,

respectively. The Euler angles vector and the force vector u is defined as:

$$\eta = \begin{bmatrix} \psi \\ \theta \\ \phi \end{bmatrix} \quad u = \begin{bmatrix} u_x \\ u_y \\ u_z \end{bmatrix} \tag{B.3}$$

Then, from (3.24), (B.1), (B.2) and (B.3) it can be deduced the thrust and the Euler angles needed to generate the virtual control u^{υ}. The ψ_{ref} needed can be chosen arbitrarily or conveniently. θ_{ref}, ϕ_{ref} and f_{ref} have the following expressions:

$$\theta_{ref} = \arctan\left[\frac{u_y s\psi + u_x c\psi}{u_z + mg}\right] \tag{B.4}$$

$$\phi_{ref} = \arctan\left[c\theta_{ref} \cdot \frac{u_x s\psi - u_y c\psi}{u_z + mg}\right] \tag{B.5}$$

$$f_{ref} = \frac{u_z + mg}{c\theta_{ref} \cdot c\phi_{ref}} \tag{B.6}$$

Let us define the Euler angles error as:

$$e_\eta = \eta - \eta_{ref} \quad \Longrightarrow \quad \begin{array}{rcl} \dot{e}_\eta & = & \dot{\eta} - \dot{\eta}_{ref} \\ & = & B(\eta)\Omega - \dot{\eta}_{ref} \end{array} \tag{B.7}$$

The matrix $B(\eta)$ has the following form:

$$B = \begin{bmatrix} 0 & c\phi & -s\phi \\ 0 & s\phi/c\theta & c\phi/c\theta \\ 1 & s\phi \cdot t\theta & c\phi \cdot t\theta \end{bmatrix} \tag{B.8}$$

where t means $tan(*)$. The matrix $B(\eta)$ is not singular if and only if $\cos(\theta) \neq 0$.

Let us propose the next Lyapunov function

$$V_{L\eta} = \frac{1}{2} < \chi_2, K_{I\eta}\chi_2 > + \frac{1}{2} < e_\eta, e_\eta > \tag{B.9}$$

where,

$$\chi_2 = \int_0^t e_\Psi \; d\tau \tag{B.10}$$

and $K_{I\eta}$ is a positive diagonal constant matrix that will be used for tuning the control. Thus,

$$\dot{V}_{L\eta} = < \chi_2, K_{I\eta} e_\eta > + < e_\eta, \dot{e}_\eta > \tag{B.11}$$

and by choosing the virtual angular velocity Ω^v,

$$\Omega^v = B^{-1}\left(\dot{\eta}_{ref} - K_{I\eta}\chi_2 - K_\eta e_\eta\right) \tag{B.12}$$

with K_η as a positive diagonal constant matrix for tuning the control, it yields

$$V_{L\eta} = -K_\eta < e_\eta, e_\eta > \quad \leq 0 \quad \forall\, t \geq 0 \tag{B.13}$$

Now, let us define the angular velocity error as:

$$e_\Omega = \Omega - \Omega_\eta \quad \Longrightarrow \quad \dot{e}_\Omega = \dot{\Omega} - \dot{\Omega}_\eta \tag{B.14}$$

remember that,

$$\Omega = \Omega_\eta + e_\Omega \quad \& \quad \dot{\Omega} = \mathbb{J}^{-1}\left(\tau - \omega^\times \mathbb{J}\Omega\right) \tag{B.15}$$

Now, consider the following candidate Lyapunov function:

$$V_{L\Omega} = V_{L\eta} + \frac{1}{2} < e_\Omega, e_\Omega > \tag{B.16}$$

so, then

$$\dot{V}_{L\Omega} = \dot{V}_{L\eta} + < e_\Omega, \dot{e}_\Omega > \tag{B.17}$$

thus,

$$\dot{V}_{L\Omega} = - < e_\eta, K_\eta e_\eta > + < e_\eta, B e_\Omega > + < e_\Omega, \dot{e}_\Omega > \tag{B.18}$$

and by choosing,

$$\tau = \omega^\times \mathbb{J}\Omega + \mathbb{J}\left(\dot{\Omega}_\eta - B^T e_\eta - K_\Omega e_\Omega\right) \tag{B.19}$$

it yields,

$$\dot{V}_{L\Omega} = - < e_\eta, K_\eta e_\eta > - < e_\Omega, K_\Omega e_\Omega > \quad \leq 0 \quad \forall\, t \geq 0 \tag{B.20}$$

with K_Ω as a positive diagonal constant matrix for the tuning of the control law.

B.3.2 Hardware setup

Our prototype mini-quadrotor is based on the mechanical structure developed by FLEXBOT[1]. The attitude control law is executed on Flight Control System Microwii Copter Processor ATMega32u4, Gyro and Accel. Then, a ground station estimates the position and attitude of the mini-quadrotor using the Vicon system. With this system it is possible to compute the position and attitude up to 100Hz. The estimated states are sent to MATLAB/Simulink through a UDP frame every 2ms. The position control algorithm is

1. http://www.flexbot.com

implemented in real-time at 200Hz on a computer using xPC target toolbox . The control variables are finally sent back to the mini-quadrotor on the Microwii, through a GIPSA-lab's built-in bridge that converts UDP frames to Bluetooth protocol. The helmet is a TELEPORTER by fatshark [2]. The Vicon System GIPSA-LAB ["2016"] is used to obtain the 3D position and rotation (q) of the mini-quadrotor and the helmet. A PC (simulink) is used to compute the algorithm of velocity control. The data sample of simulink is arranged to 0.01 seconds. These components can be seen in the schema of the Fig.B.2 using the Vicon system. A video showing the experimental results can be viewed at Tellez [2016].

B.3.3 Experimental test

Two experiments were performed to evaluate the behavior the control laws proposed in Chapter 5 and section B.3.1. The scenario for both experiments is as follow :

- The mini-quadrotor is flying directly using the proposed control laws
- The attitude control law is applied for both velocity control strategies proposed
- The altitude, $axis - z$, is controlled by the control law, and it is set at one meter
- The velocity of $axis - x$ and $axis - y$ is controlled by the reference inputs
- The reference inputs are acquired by the helmet orientation $[\phi_{head}, \theta_{head}, \psi_{head}]$. Where the orientation is obtained for the head movements. Random tilt movements are did in order to shift the system in the $axis - x$ and $axis - y$.

The specification and parameters of the mini-quadrotor prototype are given in the Table B.1.

Table B.1 – The specification and parameters of the Quad-rotor

Parameter	Description	Value	Units
m	Mass	0.057	Kg
d	Distance	0.042	m
J_x	Inertia in x-axis	0.0006833	Kg· m^2
J_y	Inertia in y-axis	0.0006833	Kg· m^2
J_z	Inertia in z-axis	0.00042993	Kg· m^2

2. http://www.fatshark.com/

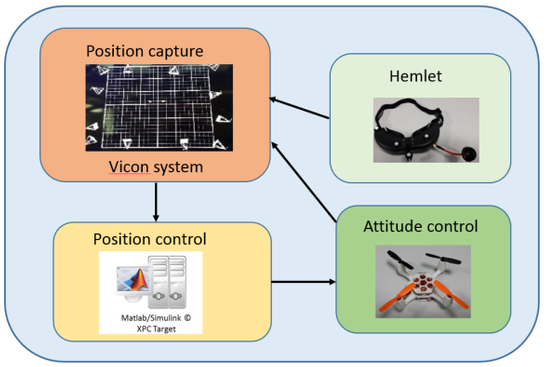


Figure B.2 – Control System

Attitude boundary control

The parameters of the control law are selected according to the characteristics of the actuators and those of the quad-rotor presented above. For the attitude control given in section (5.6), the maximum torque is $\Gamma = 0.085Nm$. Thus, the saturation function is set as $\sigma_{123} = 0.04$, and the parameters are given as $k_{12} = 0.094$, $k_3 = 0.15$,$\rho_{12} = 0.022$ and $\rho_3 = 0.035$.

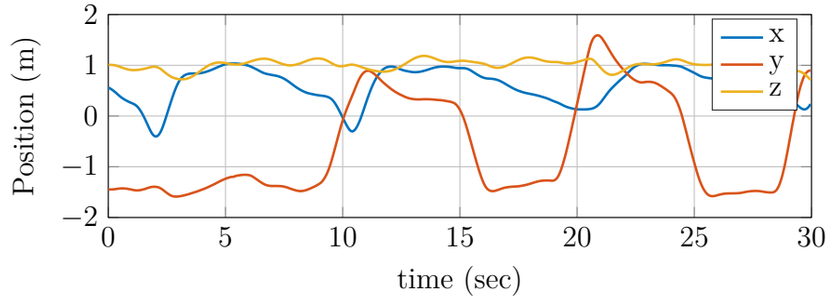
Test: Control law for trajectory tracking

The second test is shown in Fig. B.3. The parameters of the control are $K_\eta = diag[4, 4, 4]$ and $K_\Omega = diag[1, 1, 1]$.

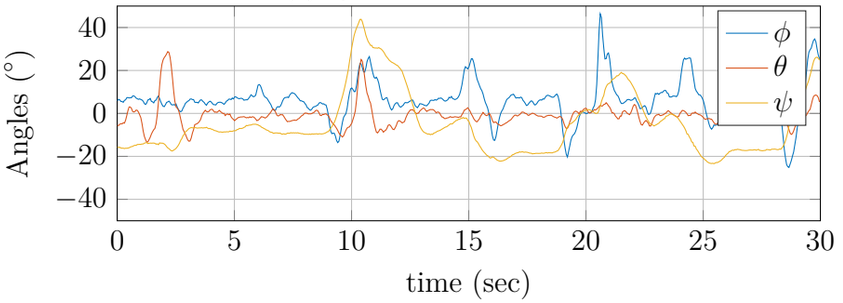
The movement of the system though $axis{-}x$ can be seen in second 2, see Fig. B.3a, and the movement is taking in account the displacement of the head movement represented at the same time in Fig. B.3d. Nevertheless, there is some delay between the real velocity and the reference velocity, that can be seen comparing both signal data from Fig. B.3d. The same behavior for $axis - y$ can be observed in second 6, see Fig. B.3a and Fig. B.3c. A delay of 0.25 seg of the reference of the velocity can be observed in both Fig. B.3c and Fig. B.3c. Fig. B.3b shows the Euler angles where the maximum angle, from ϕ and θ, is around $+ -$ 45. In this test the displacements are a bit aggressive. The ψ is controlled by the turn of the head in $axis - z$. .

Test: Velocity boundary control

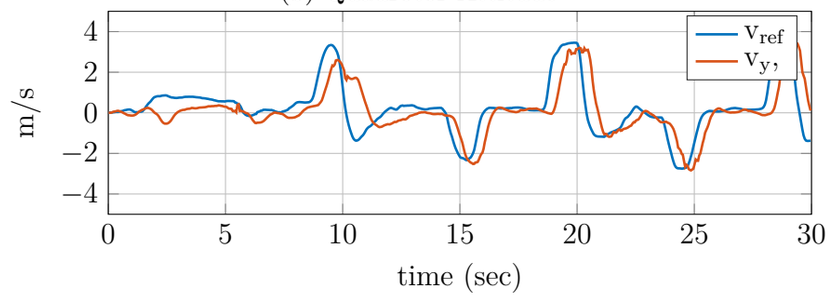
The parameters for the control law , section5.3, are $a_1 = a_2 = 1$ and $\bar{r}_1 = 1$,$\bar{r}_2 = 1$, $\bar{r}_3 = 5$. Fig. B.4a show the position of the $axis - z$, which is set at one meter. At the same time, the desplacement of the $axis - x$ and $axis - y$ can be seen in Fig. B.4a. Fig. B.4c and Fig. B.4d show the reference velocity V_{ref} and velocity from axes $[x, y]^T$. The euler angles



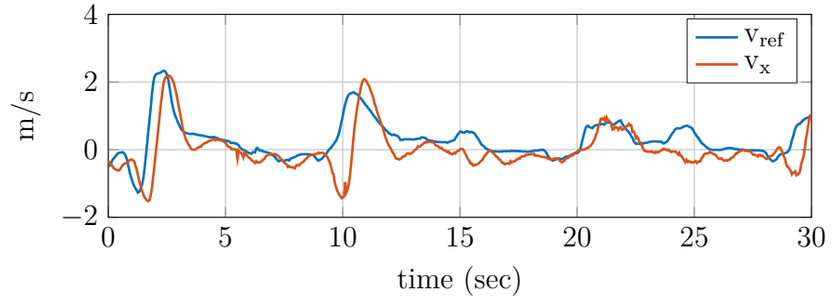
(a) Quadrotor Position



(b) Quadrotor Attitude



(c) ϕ_{head} and v_y



(d) θ_{head} and v_x

Figure B.3 – Backstepping Velocity Control with head movement

are depicted in Fig. B.4b.

The test, detailed above, consists on the displacement of the mini quad-rotor using the head movements. This movement can be seen in second 4 for the $axis - y$ in Fig. B.4a, and the movement is taking in acount the displacement of the head movement represented at the same time in Fig. B.4d. Nevertheless, there is some delay between the real velocity and the reference velocity, that can be seen comparing both signal data from Fig. B.4d. The same behavior for $axis - x$ can be observed in second 6, see Fig. B.4a and Fig. B.4c. Fig. B.4b shows the euler angles where the maximum angle, from ϕ and θ, is around $+ - 30$. That means that the displacements are slow and smooth. ψ is fixed to zero.

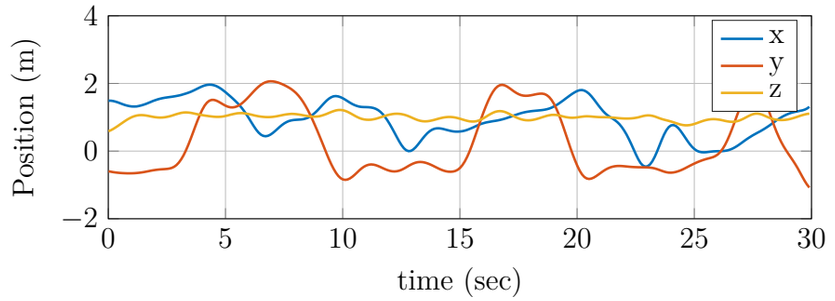
B.4 conclusions

This work proposes a teleoperation scheme to control a mini-quadrotor using only head movements from an operator. Quad-rotor position, quad-rotor orientation, and helmet movements are obtained using the Vicon System. Two velocity control algorithms were proven in order to observer the feasibility between these two control laws.

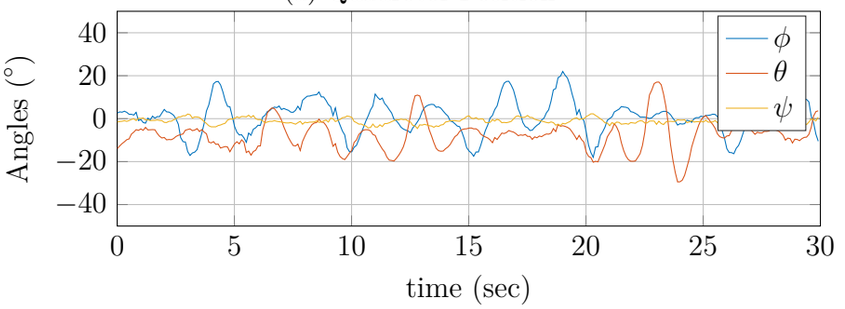
The behavior from both algorithms are quite similar; however there are some differences. For example, algorithm from section 5.4 is soft and slow to the input references, which depend on the movements of the operator's head, but algorithm from section B.3.1 is more reactive and accurate for the follow-up of the input references. In addition, the second algorithm can also command the yaw angle ψ when the operator rotates the head.

Both strategies can be used to perform the tele-operation. However, 5.4 can gives us a soft control, since using the right parameters this strategy can be used to do soft movements that can help inexperienced operators.

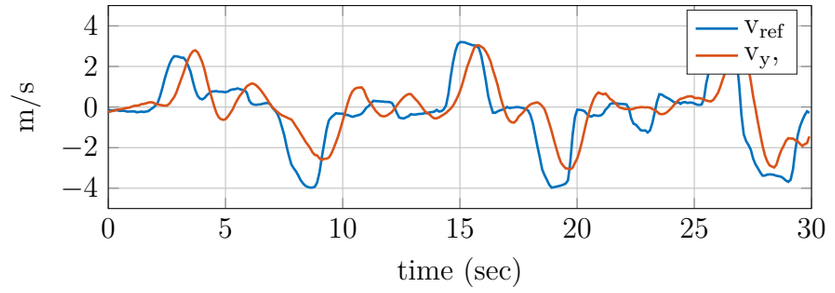
Finally, the manipulation of the mini-quadrotor with this method is more intuitive than using a radio control. When the system FPV is used as part of the teleo-peration, the manipulation is difficult, because the perception of the environment is difficult coming to a sense of disorientation. So, a new strategy is presented along this work to help the tele-operation of the systems. The tele-operation is planned to be made in indoor environment, particularly into a corridor.



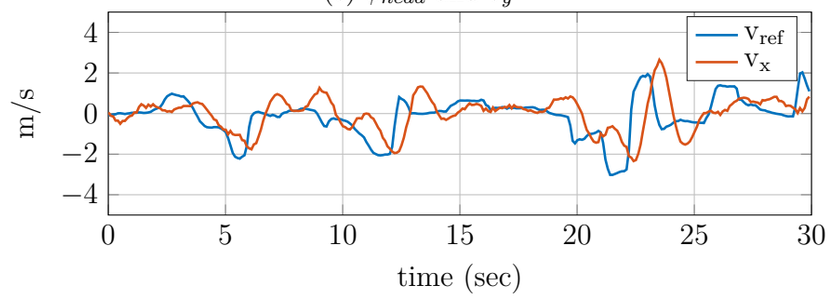
(a) Quadrotor Position



(b) Quadrotor Attitude



(c) ϕ_{head} and v_y



(d) θ_{head} and v_x

Figure B.4 – Boundary Velocity Control with head movement

Bibliography

C. Akinlar and C. Topal. Edlines: Real-time line segment detection by edge drawing (ed). In *2011 18th IEEE International Conference on Image Processing*, pages 2837–2840, Sept 2011. doi: 10.1109/ICIP.2011.6116138.

J. Alvarez-Munoz, N. Marchand, J. F. Guerrero-Castellanos, J. J. Tellez-Guzman, J. Escareno, and M. Rakotondrabe. Rotorcraft with a 3dof rigid manipulator: Quaternion-based modeling and real-time control tolerant to multi-body couplings. *International Journal of Automation and Computing*, 15(5):547–558, Oct 2018. ISSN 1751-8520. doi: 10.1007/s11633-018-1145-8.

J. U. Alvarez-Muñoz, N. Marchand, J. F. Guerrero-Castellanos, A. E. López-Luna, J. J. Téllez-Guzmán, J. Colmenares-Vazquez, S. Durand, J. Dumon, and G. Hasan. Nonlinear control of a nano-hexacopter carrying a manipulator arm. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4016–4021, Sep. 2015.

M. E. Antone and S. Teller. Automatic recovery of relative camera rotations for urban scenes. In *Proceedings IEEE Conference on Computer Vision and Pattern Recognition. CVPR 2000 (Cat. No.PR00662)*, volume 2, pages 282–289 vol.2, June 2000.

Stephen T. Barnard. Interpreting perspective images. *Artif. Intell.*, 21(4):435–462, November 1983. ISSN 0004-3702. doi: 10.1016/S0004-3702(83)80021-6. URL `http://dx.doi.org/10.1016/S0004-3702(83)80021-6`.

Cooper Bills, Joyce Chen, and Ashutosh Saxena. Autonomous mav flight in indoor environments using single image perspective cues. In *In Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA*, pages 5776–5783, 2011.

K Boulanger, K. Bouatouch, and S. Pattanaik. Atip: A tool for 3d navigation inside a single image with automatic camera calibration. *EG UK Theory and Practice of Computer Graphics*, 2006a.

Kevin Boulanger, Kadi Bouatouch, and Sumant Pattanaik. ATIP: A Tool for 3D Navigation inside a Single Image with Automatic Camera Calibration. In EUROGRAPHICS, editor, *EG UK conference*, Middlesbrough, United Kingdom, June 2006b. EUROGRAPHICS, WILEY. URL https://hal.inria.fr/inria-00461526.

Pierre-Jean Bristeau, François Callou, David Vissiére, and Nicolas Petit. The navigation and control technology inside the ar.drone micro uav. *IFAC Proceedings Volumes*, 44(1):1477 – 1484, 2011. ISSN 1474-6670. doi: https://doi.org/10.3182/20110828-6-IT-1002.02327. URL http://www.sciencedirect.com/science/article/pii/S1474667016438188. 18th IFAC World Congress.

B. Caprile and V. Torre. Using vanishing points for camera calibration. *International Journal of Computer Vision*, 4(2):127–139, Mar 1990a. ISSN 1573-1405. doi: 10.1007/BF00127813. URL https://doi.org/10.1007/BF00127813.

Bruno Caprile and Vincent Torre. Using vanishing points for camera calibration. *International Journal of Computer Vision*, 4:127–139, 1990b.

P. Castillo, A. Dzul, and R. Lozano. Real-time stabilization and tracking of a four rotor mini rotorcraft. In *IEEE Transactions on Control Systems Technology*, volume 12, pages 510–516, 2004.

D. Catuhe and D. Rousset. BABYLONJS, 2016. URL http://www.babylonjs.com/.

K. Celik, Soon-Jo Chung, and A. Somani. Mono-vision corner slam for indoor navigation. In *2008 IEEE International Conference on Electro/Information Technology*, pages 343–348, May 2008. doi: 10.1109/EIT.2008.4554326.

R. T. Collins and R. S. Weiss. Vanishing point calculation as a statistical inference on the unit sphere. In *[1990] Proceedings Third International Conference on Computer Vision*, pages 400–403, Dec 1990. doi: 10.1109/ICCV.1990.139560.

J. Colmenares-Vázquez, N. Marchand, P. Castillo, J. E. Gómez-Balderas, J. U. Alvarez-Muñoz, and J. J. Téllez-Guzmán. Integral backstepping control for trajectory tracking of a hybrid vehicle. In *2015 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 209–217, June 2015a.

J. Colmenares-Vázquez, N. Marchand, J. E. Gómez-Balderas, P. Castillo, J. J. Téllez-Guzmán, J. U. Álvarez-Muñoz, and J. Dumon. Position control of a quadrotor under external constant disturbance. In *2015 Workshop on Research, Education and Development of Unmanned Aerial Systems (RED-UAS)*, pages 180–185, Nov 2015b. doi: 10.1109/RED-UAS.2015.7441005.

J. M. Coughlan and A. L. Yuille. Manhattan world: compass direction from a single image by bayesian inference. In *Proceedings of the Seventh IEEE International Conference on Computer Vision*, volume 2, pages 941–947 vol.2, 1999. doi: 10.1109/ICCV.1999.790349.

R. Cruz-José, J. F. Guerrero-Castellanos, W. F. Guerrero-Sánchez, and J. J. Oliveros-Oliveros. Estabilización global de mini naves aéreas tipo vtol. In *Congreso Nacional de Control Automático*, Campeche, México, 2012.

Richard O. Duda and Peter E. Hart. Use of the hough transformation to detect lines and curves in pictures. *Commun. ACM*, 15(1):11–15, January 1972a. ISSN 0001-0782. doi: 10.1145/361237.361242. URL http://doi.acm.org/10.1145/361237.361242.

Richard O. Duda and Peter E. Hart. Use of the hough transformation to detect lines and curves in pictures. *Commun. ACM*, 15(1):11–15, January 1972b. ISSN 0001-0782. doi: 10.1145/361237.361242. URL http://doi.acm.org/10.1145/361237.361242.

Wael Elloumi, Sylvie Treuillet, and Rémy Leconge. Tracking Orthogonal Vanishing Points in Video Sequences for a Reliable Camera Orientation in Manhattan World. In *International Congress on Image and Signal Processing (CISP 2012)*, page 5, Chongqing, China, October 2012. URL https://hal.archives-ouvertes.fr/hal-00771525.

Wael Elloumi, Sylvie Treuillet, and Rémy Leconge. Real-time camera orientation estimation based on vanishing point tracking under manhattan world assumption. *Journal of Real-Time Image Processing*, pages 1–16, 2014. ISSN 1861-8219. doi: 10.1007/s11554-014-0419-9. URL http://dx.doi.org/10.1007/s11554-014-0419-9.

N. Eric and J. Jang. Kinect depth sensor for computer vision applications in autonomous vehicles. In *2017 Ninth International Conference on Ubiquitous and Future Networks (ICUFN)*, pages 531–535, July 2017. doi: 10.1109/ICUFN.2017.7993842.

T. I. Fossen. *Guidance and Control of Ocean Vehicles*. John Wiley and Sons, New York, 1994.

Luis Rodolfo García Carrillo, Alejandro Enrique Dzul López, Rogelio Lozano, and Claude Pégard. Combining stereo vision and inertial navigation system for a quad-rotor uav. *Journal of Intelligent & Robotic Systems*, 65(1):373–387, Jan 2012. doi: 10.1007/s10846-011-9571-7. URL https://doi.org/10.1007/s10846-011-9571-7.

GIPSA-LAB. MOCA Platform, "2016". URL http://www.gipsa-lab.grenoble-inp.fr/recherche/plates-formes.php?id_plateforme=79.

S. B. Gokturk, H. Yalcin, and C. Bamji. A time-of-flight depth sensor - system description, issues and solutions. In *2004 Conference on Computer Vision and Pattern Recognition Workshop*, pages 35–35, June 2004. doi: 10.1109/CVPR.2004.291.

Jose-Ernesto Gomez-Balderas, Pedro Castillo, Jose Alfredo Guerrero, and Rogelio Lozano. Vision based tracking for a quadrotor using vanishing points. *Journal of Intelligent & Robotic Systems*, 65(1):361–371, 2011. ISSN 1573-0409. doi: 10.1007/s10846-011-9579-z. URL http://dx.doi.org/10.1007/s10846-011-9579-z.

J.F. Guerrero-Castellanos, N. Marchand, A. Hably, S. Lesecq, and J. Delamare. Bounded attitude control of rigid bodies: Real-time experimentation to a quadrotor mini-helicopter. *Control Engineering Practice*, 19(8):790–797, 2011a.

J.F. Guerrero-Castellanos, N. Marchand, A. Hably, S. Lesecq, and J. Delamare. Bounded attitude control of rigid bodies: Real-time experimentation to a quadrotor mini-helicopter. *Control Engineering Practice*, 19(8):790 – 797, 2011b. ISSN 0967-0661. doi: https://doi.org/10.1016/j.conengprac.2011.04.004. URL http://www.sciencedirect.com/science/article/pii/S0967066111000700.

P. V. C. Hough. Machine Analysis of Bubble Chamber Pictures. *Conf. Proc.*, C590914: 554–558, 1959.

D. C. Lee, M. Hebert, and T. Kanade. Geometric reasoning for single image structure recovery. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 2136–2143, June 2009. doi: 10.1109/CVPR.2009.5206872.

Jeong-Kyun Lee and Kuk-Jin Yoon. Real-time joint estimation of camera orientation and vanishing points. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1866–1874, June 2015.

Bo Li, Kun Peng, Xianghua Ying, and Hongbin Zha. Simultaneous vanishing point detection and camera calibration from single images. In George Bebis, Richard Boyle, Bahram Parvin, Darko Koracin, Ronald Chung, Riad Hammound, Muhammad Hussain, Tan Kar-Han, Roger Crawfis, Daniel Thalmann, David Kao, and Lisa Avila, editors, *Advances in Visual Computing*, pages 151–160, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg. ISBN 978-3-642-17274-8.

Bo Li, Kun Peng, Xianghua Ying, and Hongbin Zha. Vanishing point detection using cascaded 1d hough transform from single images. *Pattern Recognition Letters*, 33(1): 1 – 8, 2012. ISSN 0167-8655. doi: https://doi.org/10.1016/j.patrec.2011.09.027. URL http://www.sciencedirect.com/science/article/pii/S0167865511003060.

K. McGuire, G. de Croon, C. De Wagter, K. Tuyls, and H. Kappen. Efficient optical flow and stereo vision for velocity estimation and obstacle avoidance on an autonomous pocket drone. *IEEE Robotics and Automation Letters*, 2(2):1070–1076, April 2017. ISSN 2377-3766. doi: 10.1109/LRA.2017.2658940.

I. F. Mondragón, P. Campoy, C. Martínez, and M. A. Olivares-Méndez. 3d pose estimation based on planar object tracking for uavs control. In *2010 IEEE International Conference on Robotics and Automation*, pages 35–41, May 2010. doi: 10.1109/ROBOT.2010.5509287.

R. C. Nelson. *Flight Stability and Automatic Control*, volume second edition. McGraw-Hill, Boston, 1998.

Marcos Nieto and Luis Salgado. Simultaneous estimation of vanishing points and their converging lines using the em algorithm. *Pattern Recognition Letters*, 32(14):1691 – 1700, 2011. ISSN 0167-8655. doi: https://doi.org/10.1016/j.patrec.2011.07.018. URL `http://www.sciencedirect.com/science/article/pii/S016786551100239X`.

R. Orghidan, J. Salvi, M. Gordan, and B. Orza. Camera calibration using two or three vanishing points. In *2012 Federated Conference on Computer Science and Information Systems (FedCSIS)*, pages 123–130, Sept 2012.

Ram Prasad Padhy, Feng Xia, Suman Kumar Choudhury, Pankaj Kumar Sa, and Sambit Bakshi. Monocular vision aided autonomous uav navigation in indoor corridor environments. *IEEE Transactions on Sustainable Computing*, 4:96–108, 2019.

Viorica Pătrăucean, Pierre Gurdjos, and Rafael Grompone von Gioi. A parameterless line segment and elliptical arc detector with enhanced ellipse fitting. In Andrew Fitzgibbon, Svetlana Lazebnik, Pietro Perona, Yoichi Sato, and Cordelia Schmid, editors, *Computer Vision – ECCV 2012*, pages 572–585, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg. ISBN 978-3-642-33709-3.

K. Schauwecker and A. Zell. On-board dual-stereo-vision for autonomous quadrotor navigation. In *2013 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 333–342, May 2013. doi: 10.1109/ICUAS.2013.6564706.

Simon, Davide Scaramuzza, Stephan Weiss, and Roland Siegwart. Mav navigation through indoor corridors using optical flow. In *Proceedings - IEEE International Conference on Robotics and Automation*, pages 3361–3368, 05 2010.

H. J. Sussmann, E. D. Sontag, and Y. Yang. A general result on the stabilization of linear systems using bounded controls. *IEEE Transactions on Automatic Control*, 39 (12):2411–2425, Dec 1994. ISSN 0018-9286. doi: 10.1109/9.362853.

Andrew R. Teel. Global stabilization and restricted tracking for multiple integrators with bounded controls. *Systems & Control Letters*, 18(3):165 – 171, 1992. ISSN 0167-6911. doi: https://doi.org/10.1016/0167-6911(92)90001-9. URL `http://www.sciencedirect.com/science/article/pii/0167691192900019`.

Tellez. Video, 2016. URL `https://drive.google.com/file/d/0B1SyiY3YGK48Q0tJVUpXdkhTSkE/view?usp=sharing`. Version 1.0.

TELLEZ. Video, 2018. URL `https://drive.google.com/file/d/1RsmXx7GODY_kSlWZZwvFZzNMKlRRXG0-/view?usp=sharing`.

J. J. Tellez-Guzman, J. E. Gomez-Balderas, N. Marchand, P. Castillo, J. C. Vazquez, J. U. Alvarez-Munoz, and J. Dumon. Velocity control of mini-uav using a helmet system. In *2015 Workshop on Research, Education and Development of Unmanned Aerial Systems (RED-UAS)*, pages 329–335, Nov 2015. doi: 10.1109/RED-UAS.2015.7441024.

Cihan Topal and Cuneyt Akinlar. Edge drawing: A combined real-time edge and segment detector. *J. Vis. Comun. Image Represent.*, 23(6):862–872, August 2012. ISSN 1047-3203. doi: 10.1016/j.jvcir.2012.05.004.

P. Vadakkepat, T. C. Chong, W. A. Arokiasami, and X. Weinan. Fuzzy logic controllers for navigation and control of ar. drone using microsoft kinect. In *2016 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, pages 856–863, July 2016. doi: 10.1109/FUZZ-IEEE.2016.7737778.

Fei Wang, Jin-Qiang Cui, Ben-Mei Chen, and Tong H Lee. A comprehensive uav indoor navigation system based on vision optical flow and laser fastslam. *Acta Automatica Sinica*, 39(11):1889 – 1899, 2013. ISSN 1874-1029. doi: https://doi.org/10.1016/S1874-1029(13)60080-4. URL `http://www.sciencedirect.com/science/article/pii/S1874102913600804`.

Y Wang. An efficient algorithm for uav indoor pose estimation using vanishing geometry. *Proceedings of the 12th IAPR Conference on Machine Vision Applications, MVA 2011*, pages 361–364, 01 2011.

Stephan Weiss, Davide Scaramuzza, and Roland Siegwart. Monocular-slam-based navigation for autonomous micro helicopters in gps-denied environments. *Journal of*

Field Robotics, 28(6):854–874, 2011. ISSN 1556-4967. doi: 10.1002/rob.20412. URL http://dx.doi.org/10.1002/rob.20412.

A. R. Ximenes, P. Padmanabhan, M. Lee, Y. Yamashita, D. N. Yaung, and E. Charbon. A 256×256 45/65nm 3d-stacked spad-based direct tof image sensor for lidar applications with optical polar modulation for up to 18.6db interference suppression. In *2018 IEEE International Solid - State Circuits Conference - (ISSCC)*, pages 96–98, Feb 2018. doi: 10.1109/ISSCC.2018.8310201.

X. Zhang, K. Wang, Z. Zhang, and Q. Yu. A new line-based orthogonal iteration pose estimation algorithm. In *2009 International Conference on Information Engineering and Computer Science*, pages 1–4, Dec 2009. doi: 10.1109/ICIECS.2009.5366007.

Z. Zhang. A flexible new technique for camera calibration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(11):1330–1334, Nov 2000. ISSN 0162-8828. doi: 10.1109/34.888718.