Towards safe robot arm motion close to humans

Pu Zheng

THÈSE

Pour obtenir le grade de

DOCTEUR DE L'UNIVERSITÉ GRENOBLE ALPES

École doctorale : MSTII - Mathématiques, Sciences et technologies de l'information, Informatique
Spécialité : Informatique
Unité de recherche : Laboratoire d'Informatique de Grenoble

Vers des mouvements sûrs pour un bras robotique opérant à proximité d'humains

Towards safe robot arm motion close to humans

Présentée par :

Pu ZHENG

Direction de thèse :

Olivier AYCARD MCF, UGA	Directeur de thèse
Pierre-Brice WIEBER CHARGE DE RECHERCHE, Université Grenoble Alpes	Co-encadrant de thèse

Rapporteurs :

VINCENT PADOIS
Directeur de recherche, INRIA CENTRE BORDEAUX SUD-OUEST
OLIVIER STASSE
Directeur de recherche, CNRS DELEGATION OCCITANIE OUEST

Thèse soutenue publiquement le **12 décembre 2022**, devant le jury composé de :

OLIVIER AYCARD Maître de conférences HDR, UNIVERSITE GRENOBLE ALPES	Directeur de thèse
VINCENT PADOIS Directeur de recherche, INRIA CENTRE BORDEAUX SUD-OUEST	Rapporteur
OLIVIER STASSE Directeur de recherche, CNRS DELEGATION OCCITANIE OUEST	Rapporteur
MASSIH-REZA AMINI Professeur des Universités, UNIVERSITE GRENOBLE ALPES	Président
PAUL CHECCHIN Professeur des Universités, UNIVERSITE CLERMONT- FERRAND 1 - AUVERGNE	Examinateur

Invités :

PIERRE-BRICE WIEBER
Chargé de recherche, INRIA CENTRE GRENOBLE-RHONE-ALPES

Abstract

Collaborative robots offer new possibilities to use robots in workspaces shared with humans. These robots can interact with their environment and assist human beings in their task in a safer way compared to standard industrial ones. They are required to be fast, precise, and efficient during the accomplishment of their tasks. However, their strength can make them dangerous tools around people. Therefore, to ensure safety they are often used in a sub-optimal way.

The aim of this work is to ensure the safety of a human interacting with a robot performing a set of tasks. It is particularly focused on the generation of collision-free trajectories. Online analysis of the current human motion is not sufficient to guarantee a fluent and safe interaction. Instead, the robot needs to be able to anticipate human motion to be able to initiate actions in time and to render long–term planning possible. These robots are also limited by their intrinsic design. However, it's impossible to guarantee that no collision will ever take place in a partially unknown dynamic environment such as a shared workspace. In this situation, no matter what the future trajectory of the robot is, a collision eventually occurs. But we can guarantee instead that, if a collision takes place, the robot is at rest at the time of collision, so that it doesn't inject its own kinetic energy in the collision.

To that aim, this work formulates the planning problem as a constrained optimization one under Model Predictive Control framework and solves it using Linear Quadratic Programming. The collision-free constraint is based on a separating plane method to calculate the distance between the robot and the person. As we anticipate the movement of the person so that the robot avoids in advance when there is a potential collision, we need therefore to predict its movement. The prediction is performed by neural networks which are efficient in predicting sequential data and can learn certain motion patterns.

The proposed method is validated by implementing them for online trajectory generation on a 7 dof serial robotic manipulator. This robot is used in a shared workspace with a human. Using external sensors it is shown that it is possible to realise tasks while reactively avoiding a potential collision.

Keywords: Human/Robot Collaboration, Model Predictive Control, Safety, Quadratic Programming, Redundant Robots, Motion Prediction, Deep Learning, Motion Planning

Contents

Nomenclature

Abbreviations

$C-space$	Configuration space
Dof	Degree of freedom
EE	End effector
HRC	Human robot collaboration
HRI	Human robot interaction
MPC	Model predictive control
QP	Quadratic programming

Symbols and Notation

$\overline{u}$	Upper bound of the control vector
$\mathbf{u}$	$\{u^0, u^1, \ldots, u^{N-1}\}$ is the predicted input vectors on control horizon
$\mathbf{x}$	$\{x^1, x^2, \ldots, x^N\}$ is the predicted state vectors on control horizon
$\underline{\mathrm{u}}$	Lower bound of the control vector
$\{p^1, p^2, \ldots\}$	Vertices of a polyhedra which characterise a person
$\{r^1, r^2, \ldots\}$	Vertices of a polyhedra which characterise a robot
A	$A \in \mathbb{R}^{2n \times 2n}$ is the state transition matrix of linear dynamic system
a	Normal vector of a separating plane
b	Closest distance from the origin to the separating plane
C_{dq}	Inequalities constraints matrix for a quadratic programming problem associated to joint velocity
C_q	Inequalities constraints matrix for a quadratic programming problem associated to joint configuration
I	$n \times n$ identity matrix
J	The Jacobian of the robot's end-effector expressed in the robot base frame
lb_{dq}	Lower bound of the inequalities constraints for a quadratic programming problem associated to joint velocity
lb_q	Lower bound of the inequalities constraints for a quadratic programming problem associated to joint configuration

N	The total sampling steps on prediction/control horizon
u	Control vector, here is the piece-wise constant acceleration
ub_{dq}	Upper bound of the inequalities constraints for a quadratic programming problem associated to joint velocity
ub_q	Upper bound of the inequalities constraints for a quadratic programming problem associated to joint configuration
x	Robot state configuration including joint position and joint velocity
x^{des}	Desired robot state
Δt	Time sampling interval
k	Time sampling at step k
q	Robot joint configuration

List of Figures

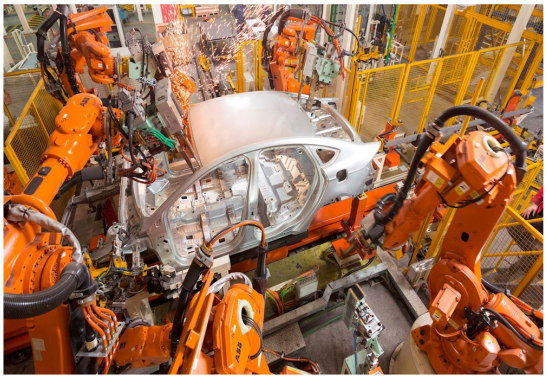
List of Tables

1 Introduction

Contents

Industrial robot begin their root in the mid of 20st-century. These automatic device are aim to help humans work for heavy, dangerous and monotonous task. According to ISO 8373:2012,[1] industrial robot is "an automatically controlled, reprogrammable multipurpose manipulator, which can be either fixed or mobile for use in industrial automation application". While the first industrial robot "Unimate" was deployed in 1962 for manufacturing, they was only able to accomplish simple pick and place task. Then, in 1980s, industrial robot starts greatly expanded to many complex tasks such as welding, painting, assembly because of the advance in mechanics, electronics and computer software. Most industrial applications of robot manipulators use position control, the performance of the robot is ensured by using a high control gain which lead the robot to become very stiff.

While in a free space stiffness is a desired behaviour, but this can lead to potential danger for a human operator working nearby and not anticipating the movement of the robot. In 1987, a study about cause-and-effect analysis of robot accidents is reported by [Jiang1987]. This study concludes that the most of accident happens during human-robot interaction when the robot is in operational mode, and it could be avoid with proper safety design during robot installation.



Source: ABB website

(a)

Source: Industries : faites le choix de la robotique collaborative humarobotic website

(b)

Figure 1.1: (a) Robots in a car assembly line working in a workspace separated by cages. The environment is perfectly know, the trajectories can be generated off-line. (b) Interaction between human and robot in a shared workspace, robot movements are restricted because the operation is moving around the robot. It should update it's trajectory on-line to adapt to the change of environment.

The trivial solution is to enclose robot in a protective barriers as shown in Figure 1.1(a) to ensure the safety. In this separating cage, the environment is static, no external perturbation can happen, the robot is able to perfectly execute a trajectory calculated offline. When the operator enters to the cage, he must first turn off the robot power supply, and then restart the robot once he is out. The robot becomes ineffective if this action happens frequently. Furthermore, they are not flexible at all for various tasks, and are therefore more used for repetitive and long periodic tasks. It is also time consuming to install such robot devices, for example, it takes 200 hours to program (and reprogram) them in an automobile industry [Djuric2016]. In addition, building a production line with industrial robots is very expensive. For all these reasons, industrial robots remain only a powerful tool for large companies, and do not at all favour small and medium-sized companies.

[1] https://www.iso.org/obp/ui/#iso:std:iso:8373:ed-2:v1:en

1.1 Towards collaborative robots in shared spaces

Collaborative robots		Conventional industry robots
Simply programming		Programming is time- consuming
Lower weigh of robot < 29kg		High weigh of robot > 50kg
Embedded security features		Missing of safety sensors
Operation in the limited space		Large operation space
Lower weigh of machined workpiece		Greater load capacity
Mobile		Immobile
External force sensors		Sensors of external forces are missing
Faster and simpler setting		High operation speed in operation
Flexibility of deployment		Universal usage in the limited space

Tableau 1.1: A comparison between industrial robots and collaborative robots. (source [Matúšová2019])

Separating robots with cages is an efficient way to ensure safety as it prevents any possible contact with the robot. However, the downside is that the automation process is either fully automated or fully manual. There are applications that can benefit from a close collaboration between humans and robots, such as in the context of industrial 4.0 and circular economy [Mehrpouya2019]. The best solution is to combine the strengths, precision, and speed of industrial robots with the ingenuity, judgment, and dexterity of human workers. This way, human workers can handle tasks that require flexibility, while robots handle tasks that make the best use of their strength and speed.

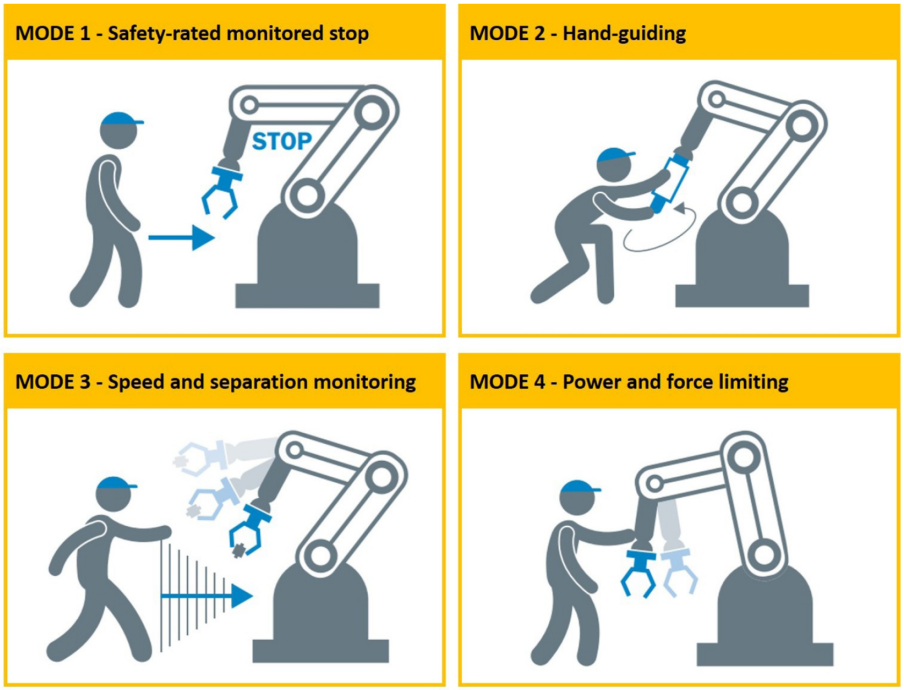
As mentioned in the previous section, the inflexibilities and dangers of traditional manipulator robots have limited their use in human-robot interaction applications. These limitations includes a lack of safety sensor, hard to displace, and high operational speeds, as summarized in Tab 1.1. To adress these issues and meet new needs such as collaborative assembly [Unhelkar2014, Bejarano2019], a home assistant has been developed to aid aging inviduals [Madarash-Hill2004]. A considerable investment is being made in companies and laboratories for the development of collaborative robots. For example, the collaborative DLR LightWeight Robot [Albu-Schäffer2007] is designed for interaction with humans in daily and unstructured environments with a load-to-weight ration of approximately 1:1. The use of lightweight materials keeps the total system weight under 15 Kg. These DLR techniques have led to a more accessible collaborative robot called Franka Emika Panda [2]. Collaborative robots have several advantages, including ease of programming for new tasks, faster and simpler set-up in new environments, and the ability to operate without the need for enclosing cages due to embedded security features. A comprehensive comparison between traditional industrial robots and collaborative robots can be found in Tab 1.1. As the table shows, the use cases for these two types of robots are very different. Traditional industrial robots are well-suited for heavy tasks and static environments, while collaborative robots are better suited for human-centered task or frequently changing assembly lines.

However, even if these collaborative robots are much less dangerous. One may still wonder how can we be sure they won't injure humans? The first document ISO 10218 entitled "Robots and robotic devices - Safety requirements for industrial robots" has been proposed to provide standardization for maintaining safety during interactions between humans and industrial robots [for Standardization2011]. The accompanying technical specification ISO/TS 15066 provides additional information and details on how to achieve the requirements established by ISO 10218 for collaborative operation. In the following section, we will delve into the different methods proposed

[2] https://www.franka.de/

by IOS 10218 and explain the motivation behind this thesis in this context.

1.2 Safety requirement in Human-Robot collaboration



source: [Magrini2020]

Figure 1.2: Four different modes of cooperation. Mode 1: Robot is fully stopped. Mode 2: Robot is passively guided by human. Mode 3: Speed reduction when human approches. Mode 4: Power and force limited

According to the technical specification ISO/TS 15066, there exists four different collaboration modes as shown in Figure1.2:

- **Safety-rated Monitored Stop**
  During interactions with operators in a shared workspace, the robot is stopped to ensure safety. This type of collaboration is well-suited for situations where the robot is assisting the operator in positionning heavy components or manually inserting object into the robot's end-effector [Vysocky2016]. A key feature of the safety-rated monitored stop is that the robot resumes movement once the operator is no longer present, eliminating the need to manually power it on again;

- **Hand-guided**
  The robot is manually guided by the operator, who acts as a teacher, guiding the robot to the desired positions by moving it directly, without the need for an intermediate programming interface. Hand-guided is often used in conjunction with Safety-rated Monitored Stop feature. When the robot is alone in shared space, it executes its automatic task. Once the operator enters, the robot stops and allows the operator to activate the hand-guided mode. A

more complete study of hand-guided, including its impact on safety, operability, and human skill assistance, is carried out by [Fujii2016];

- **Speed & Separation Monitoring** The speed and trajectory of the robot are adjusted based on the position of the person in the shared workspace. This allows the robot to continue its tasks at a reduced speed [Joseph2018] without needing to stop compeletely. Additionally, separation monitoring ensures that the robot maintais a safe distance from the operator, effectively preventing collisions [Zheng2020];

- **Power and Force Limited** Direct contact between the operator and the robot can occur intentionally and unintentionally. To ensure safety, the force that the robot can exert on the operator is limited. The technical specification ISO/TS 15066 includes maximum force values that must not be exceeded when the robot collide with body parts [Haddadin2008].

	Speed	Separation Distance	Type of Interaction
Safety-rated Monitored stop	Zero	Small	Coexistence
Hand guiding	Safety-rated monitoring speed	Small	Collaboration
Speed and separation monitoring	Safety-rated monitoring speed	Safety-rated monitoring distance	Coexistence
Power and force limiting	Determined by Limited impact forces	Small	Collaboration

Tableau 1.2: The relation between four modes and the type of interaction

During interactions between the operator and the robot, there can be direct contact or no contact at all. Therefore, we must distinguish the interaction in more specific terms: coexistence and collaboration. Coexistence refers to the robot and operator sharing a workspace but not performing the same task simultaneously. An example is an operator performing a task side by side but without having mutual contact. On the other hand, collaboration means that the robot and the operator perform a task at the same time and in the same workspace. Direct physical contact between the two is possible. The classification of the four modes of collaboration is presented in the Tab. 1.2. Depending on the type of interaction, the motion of the robot is constrained by the technical specification. Thus, we need to determine an appropriate architectural control. In this work, we don't consider physical interactions between the robot and human, such as a handing an object or holding it together, only situations where they share the same workspace and have to work separately with as little interference as possible. For this situation, the appropriate safety requirement could be define as "Speed and separation monitoring". Therefore, the robot is able to move around the operator at a safe velocity and keep a safe distance.

1.3 Motivations and contributions

1.3.1 Motivations

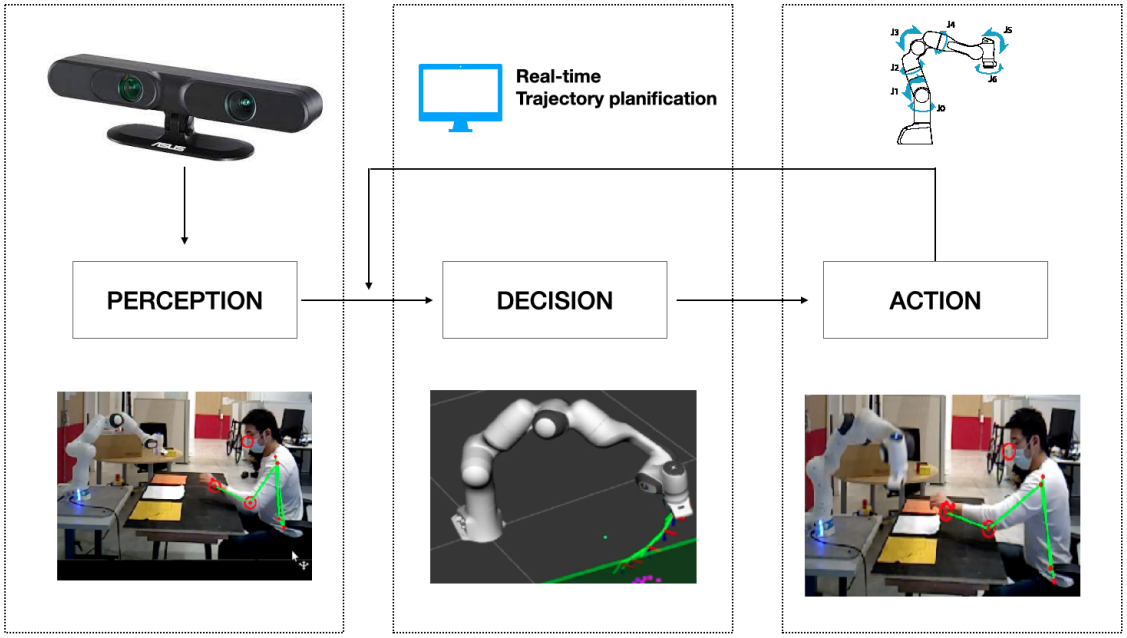


Figure 1.3: The perception-decision-action loop for robot architecture.

Close interaction and collaboration with humans in shared workspaces require robots to be able to coordinate their actions in space and time with their human partners. Online analysis of the current human motion is not sufficient to guarantee a fluent and safe interaction. Instead, the robot needs to be able to anticipate human motion, enabling them to initiate actions in time and facilitate long–term planning.

A traditional autonomous robot system is described by a perception-decision-action loop [Thrun2002]. shown in Fig 1.3. This loop can be perfectly adopted for generating collision-free trajectory during human robot collaboration:

- Perception: processing of exteroceptive sensor flows (camera) to extract information about the environment. This correspond to the first column of Fig 1.3. We localise the position of human's arm and predict the arm motion during a short horizon of time.
- Decision: from the extracted information, and taking into account the robot's objective, as well as its possibilities of motion, the robot re-generate a collision-free trajectory as shown in Figure 1.3.
- Action: performing low-level control command to follow the desired trajectory. thus the action will influence the state of its environment.

However, it's impossible to guarantee that no collision will ever take place in a partially unknown dynamic environment such as a shared workspace. In this situation, no matter what the future trajectory of the robot is, a collision eventually occurs [Fraichard2004]. But we can guarantee instead that, if a collision takes place, the robot is at rest at the time of collision, so that it doesn't inject its own kinetic energy in the collision [Bouraine2012]. This is called *passive motion safety*, and this is what we aim for.

Collision avoidance is classically implemented by restricting the motion of the robot in the direction of the human, either with a repulsive potential field [Khatib1986], velocity dampers [Faverjon1987], safety index evaluation with links' momentum projection [Tsai2014], control barrier functions that provides inequality constraints in the control input [Ames2016], invariance control [Kimmel2017]. All these approaches share two major problems. The first is that these restrictions are defined arbitrarily, only loosely related to the dynamics of the robot, and generate therefore suboptimal behaviors such as unnecessary detours or stops. The second is that they assume that the proposed restriction can always be enforced and the collision avoided, which may not be true when multiple constraints are affecting simultaneously the motion of the robot. In that case, their behavior is undefined and a collision could happen in a completely uncontrolled way.

Perceiving the environment is typically accomplished using exteroceptive sensors. For example, laser scanners can be used to survey an area and prevent workers from intruding into the robot's workspace [Ogura2012]. Other approachses, such as using a depth camera to evaluate the distance between the robot and both static and moving obstacles, have also been proposed by [Flacco2012a]. In addition, multi-depth camera fusion method have been applied by [Fabrizio2016] to reduce unobserved space. However, these approaches are limited to instantaneous detection and do not take into account the future movement of objects. Since we argue that predict and anticipate human arm's motion can improve the safety issues in human robot collaboration .

The quality of collision-free trajectory generation is highly dependent on the accuracy of our predictions. For instance, a manipulator robot moving at full speed (2 $m.s^{-1}$) can stop in approximately 0.15 s and 0.15 m [Emika]. However, in that same amount of time, a human hand starting from rest can cover more than 1 m for extreme motions such as boxing [Kimm2015a]. Therefore, if the robot is required to stop before being punched by a human, the necessary safety distance would be more than 1.15 m, which would greatly limit the possibility to sharing workspaces. However, human reaching motions are typically 10 times slower, with maximum speed approximately 1 $m.s^{-1}$ and acceleration of 5 $m.s^{-2}$ [Grujić2015a]. We propose that these slower estimates should be considered when designing the collision avoidance and passive motion safety capacities of the robot. Even if a human moves faster and a collision occurs before the robot can stop, we would still benefit from the low probability of serious injury discussed above [Albu-Schäffer2007].

The prediction of the motion of people comes in different forms: the prediction of the whole body [Martinez2017a], a single rigid body [Fridovich-Keil2020], the arms [Pereira2017a] or the hand [Ding2011a]. We will shortly describe these different approaches in this section and will elaborate on them in detail in Chapter 4. The prediction method can generally be classified into three categories: Physics-based, Planning-based, and Pattern-based [Rudenko2020]. Physics-based methods define an explicit dynamical model based on Newton's law of motion [Kakadiaris1996], [Pereira2017a], Pattern-based methods learn motion patterns from data of observed trajectories [Ding2011a], [Mainprice2013a] and Planning-based methods reason on motion intent of the person [Sylla2014b], [Mainprice2015]. These approaches will be later describe in chapter V.

1.3.2 Contributions

The contributions of this thesis are:

- We propose an predictive control (MPC) scheme for generating collision-free trajectory in real-time for a robot sharing a workspace with a human. Thus, if the collision is unavoidable, we at least ensure that the robot is at rest at the time of collision by introducing *passive motion safety* constraints.
- We introduce the "separating plane" method to calculate the distance between the robot and the person. This approach avoids the abrupt change of the pair of closest points of two objects.

- We propose to use Encoder-Decoder LSTM neural network to predict the motion of the person's arm. This prediction is included in our MPC scheme to establish collision constraints.
- We demonstrate in real experimentation this real-time collision-free trajectory generator with a robot of 7 dof (Franka Panda Emika).

1.4 Outline

here is a brief overview of the following chapters:

Chapter two: Theoretical Backgroud

Chapter 2 reviews several techniques in robotics and deep learning considered in this thesis.

Chapter three: Online motion generation in a shared workspace

Chapter 3 reviews the state of the art on motion generation of robot manipulator in the presence of obstacle. Then it introduces the general MPC scheme which is defined in three parts: i) Task definition ii) Constraints formulation and iii) Problem solving as an Quadratic Programming. The distance computing algorithm based on separating plane and *passive motion safety* will be explained and explored in this chapter.

Chapter four: Human detection and arm motion prediction

Chapter 4 reviews the state of the art on human pose detection and motion prediction based on deep learning. We construct an Encoder-Decoder LSTM (long short-term memory) neural network to predict human arm's motion and present the connection between this part and the MPC scheme.

Chapter five: Implementation and Validation

Chapter 5 validates the MPC scheme for collision-free trajectory generation in real-time presented in Chapter 3. We illustrate the case for a seven degree of freedom robot manipulator performing point-to-point operations in the presence of a dynamic human.

Chapter six: Conclusion

Chapter 6 concludes this thesis by summarising its contributions, discussing its limitations and mentioning the future developments and research directions.

2
Background

Contents

The work realized in this thesis is a result of cross research. Techniques from the different research areas contribute to the formulation and the resolution of the final motion generator. In this chapter, we provide an overview of these techniques which is important for understanding the rest of the thesis. This overview is not exhaustive and does not aim to provide a complete account of what has been done in this field. Rather, it is intended to provide the reader with enough information to situate this work with the relevant state-of-the-art approaches. In Section 2.1, we introduce the notations and definitions necessary to understand robot kinematics, such as forward kinematics and inverse kinematics. In Section 2.2, we give an overview of different trajectory planning approaches (local versus global planning) and the formulation of the optimization trajectory. Then, in Section 2.3, we provide an overview of standard constrained optimization techniques and we introduce Model Predictive Control which is closely related to optimization in Section 2.4. Finally in Section 2.5, we review some basic ideas and notations that can help readers to understand Deep Learning for computer vision systems.

2.1 Robot kinematics

Kinematics is the description of the motion (position, velocity and acceleration) of points, bodies and systems of bodies, it only describes geometrically how they move without considering the forces causing the motion. For a robot manipulator, kinematics is the study of the relationship between the robot's joint positions and its spatial layout. Kinematics is very important for many scenarios such as positioning a gripper at a location, designing a mechanism that can move a tool from point A to point B, or predicting whether a robot's motion will collide with obstacles. This section provides a brief introduction to rigid body motion in space and how it relates to robot kinematics(forward and inverse kinematics). The velocity kinematics based on the idea of Jacobian which connects the joint space velocity and the Cartesian space velocity is given at the end of this section. For a detail reference, readers are invited to consult [Spong2006, Murray2017].

2.1.1 Rigid body transformation

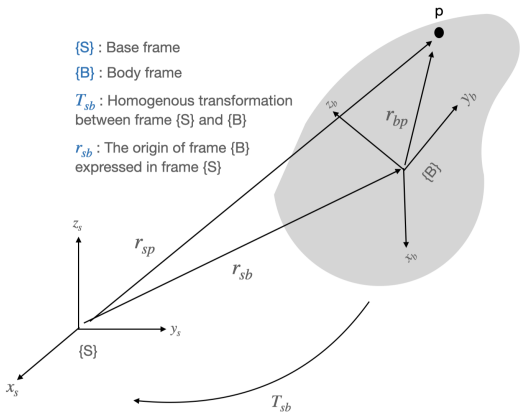


Figure 2.1: Tranformation between a body fixed frame B and base frame S

In the general case, two reference frames have a position offset and a relative rotation, as shown in Figure 2.1. As a result, the point q can be transformed from the body fixed frame B to the base frame S using a homogeneous transformation matrix T_{sb}, which is a combined translation

and rotation:

$$r_{sp} = r_{sb} + r_{bp} \tag{2.1a}$$

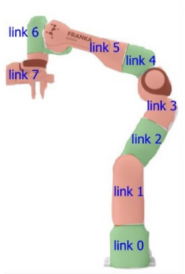
$${}_s r_{sp} = {}_s r_{sb} + R_{sb} \cdot {}_b r_{bp} \tag{2.1b}$$

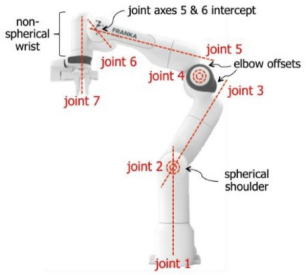
$$\begin{pmatrix} {}_s r_{sp} \\ 1 \end{pmatrix} = \underbrace{\begin{bmatrix} R_{sb} & {}_s r_{sb} \\ 0_{1\times 3} & 1 \end{bmatrix}}_{T_{sb}} \begin{pmatrix} {}_b r_{bp} \\ 1 \end{pmatrix} \tag{2.1c}$$

The Equation 2.1a gives the physical meaning between two position vectors. However, numerical calculations of vectors require them to be expressed in the same frame, in Equation 2.1b, all vectors are expressed on the base frame S through the rotation matrix R_{sb}. Combine position offset and rotation together, we obtain the homogeneous 4×4 matrix in Equation 2.1c. And consecutive homogeneous transformations are given by

$$T_{SC} = T_{SB} T_{BC} \tag{2.2}$$

2.1.2 Forward kinematics





Joint	a (m)	d (m)	α (rad)	θ (rad)
Joint 1	0	0.333	0	θ_1
Joint 2	0	0	$-\frac{\pi}{2}$	θ_2
Joint 3	0	0.316	$\frac{\pi}{2}$	θ_3
Joint 4	0.0825	0	$\frac{\pi}{2}$	θ_4
Joint 5	-0.0825	0.384	$-\frac{\pi}{2}$	θ_5
Joint 6	0	0	$\frac{\pi}{2}$	θ_6
Joint 7	0.088	0	$\frac{\pi}{2}$	θ_7
Flange	0	0.107	0	0

Figure 2.2: Geometric representation of the manipulator Franka Emika Panda and it's associated Denavit-Hartenberg parameters.

The forward kinematics problem for serial link manipulators is to determine the position and orientation of the end effector given the values of the robot's joint variables. The problem can be solved by attaching coordinate frames to each link of the robot and expressing the relationship between these frames as successive homogeneous transformations, as described previously.

The configuration of a frame requires six parameters, three for the position of the origin and three for the orientation. However, with the appropriate position of the frame in each link, only four parameters are needed to describe the configuration of a frame. this is so called Denavit-Hartenberg convention. In this convention, each homogeneous transformation T_i is represented as

a product of four basis transformations

$$T_i^{i-1}(q_i, d_i, a_i, \alpha_i) = Rot_{z,q_i} Trans_{z,d_i} Trans_{x,a_i} Rot_{x,\alpha_i} \tag{2.3a}$$

$$= \begin{bmatrix} c_{q_i} & -s_{q_i} & 0 & 0 \\ s_{q_i} & c_{q_i} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & a_i \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & a_i \\ 0 & c_{\alpha_i} & -s_{\alpha_i} & 0 \\ 0 & s_{\alpha_i} & c_{\alpha_i} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.3b}$$

$$= \begin{bmatrix} c_{q_i} & -s_{q_i} c_{\alpha_i} & s_{q_i} s_{q_i} & a_i c_{q_i} \\ s_{q_i} & c_{\theta_i} c_{\alpha_i} & -c_{q_i} s_{\alpha_i} & a_i s_{\theta_i} \\ 0 & s_{\alpha_i} & c_{\alpha_i} & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.3c}$$

Where the four quantities q_i,a_i,d_i and α_i are parameters related to link i and joint i. They are called respectively link length, link twist, link offset and joint angle. Since the matrix T_i is a function of a single variable, three of the above four quantities are constant for a given link, while the fourth parameter, q_i for a revolute joint and d_i for a prismatic joint, is the joint variable.

Therefore, the configuration of link 7 relative to the base frame (link 0) can be obtain by:

$$T_n^0 = \prod_{i=1}^{n-1} T_i^{i-1} T_{i+1}^i \tag{2.4}$$

Where n is the degree of freedom (dof), for Franka Emika Panda manipulator, $n = 7$.

2.1.3 Velocity kinematics

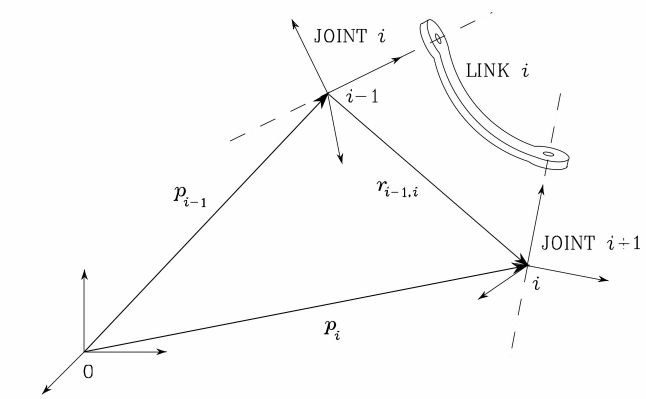


Figure 2.3: A generic link i of a serial manipulator.

Forward kinematic calculates the position and orientation of the robot end-effector frame for a given set of joint configurations. In this section, we will explore how to calculate the velocity of the end-effector of an manipulator from a give set of joint positions and velocities.

Consider a generic link i of a serial manipulator connect joints i and i+1 according to Devanit-Hartenberg convetion. Let frame i attached to link i and has origin along joint i+1 axis, while frame i-1 has origin along joint i axis as described in Figure 2.3.

Let p_{i-1} and p_i the position vectors of the origins of frames i-1 and i respect to base frame. Furthermore, let $r^{i-1}_{i-1,i}$ denote the position of the origin of the frame i with respect to frame i-1 expressed in frame i-1, we have

$$p_i = p_{i-1} + R^0_{i-1} r^{i-1}_{i-1,i} \tag{2.5}$$

Where R^0_{i-1} is the rotation matrix from frame i-1 to base frame. Taken time derivative of Equation 2.5 with respect to base frame, we obtain

$$\dot{p}_i = \dot{p}_{i-1} + R^0_{i-1} \dot{r}^{i-1}_{i-1,i} + \omega_{i-1} \times r_{i-1,i} \tag{2.6}$$

Which gives the expression of the linear velocity of link i. And the link angular velocity is given by

$$\omega_i = \omega_{i-1} + \omega_{i-1,i} \tag{2.7}$$

Where for a revolute joint, we have $\omega_{i-1,i} = \dot{q}_i z_{i-1}$ and $R^0_{i-1} \dot{r}^{i-1}_{i-1,i} = \omega_{i-1,i} \times r_{i-1,i}$. Hence, the expressions of angular velocity Equation 2.7 and linear velocity Equation 2.6 become

$$\omega_i = \omega_{i-1} + \dot{q}_i z_{i-1} \tag{2.8a}$$

$$\dot{p}_i = \dot{p}_{i-1} + \omega_i \times r_{i-1,i} \tag{2.8b}$$

Thus, the angular and linear velocities of link i as completely determined by the angular and linear velocities of the previous link i-1. This property can be efficiently used to recursively compute the velocities of the end-effector by link velocity propagation, such as

$$\dot{p}_e = \sum_{i=1}^{n} \frac{\partial p_e}{\partial q_i} \dot{q}_i = \sum_{i=1}^{n} J_{p_i} \dot{q}_i \tag{2.9a}$$

$$\omega_e = \sum_{i=1}^{n} \omega_{i-1,i} = \sum_{i=1}^{n} J_{O_i} \dot{q}_i \tag{2.9b}$$

For a revolute joint i, the contribution of the linear velocity is to be computed with reference to the origin of the end-effector frame, it is

$$J_{p_i} \dot{q}_i = \omega_{i-1,i} \times r_{i-1,e} = \dot{q}_i z_{i-1} \times (p_e - p_{i-1}) \tag{2.10}$$

Then

$$J_{p_i} = z_{i-1} \times (p_e - p_{i-1}) \tag{2.11}$$

And for angular contribution, we have

$$\dot{q}_i J_{O_i} = \dot{q}_i z_{i-1} \tag{2.12}$$

Then

$$J_{O_i} = z_{i-1} \tag{2.13}$$

Therefore, we obtain the general form Jacobian of a n-dof serial manipulator as

$$J_q = \begin{bmatrix} z_0 \times (p_e - p_0) & \cdots & z_{n-1} \times (p_e - p_{n-1}) \\ z_0 & \cdots & z_{n-1} \end{bmatrix} \tag{2.14}$$

This Jacobian is called geometric Jacobian, which is the stacked angular and linear velocity components.

The Jacobian matrix is extremely useful in robotic applications, from planification to robot control. We will exploire how to use the Jacobian to map the Cartesian configuration of obstacles into joint space in Chapter 3.

2.1.4 Inverse kinematics

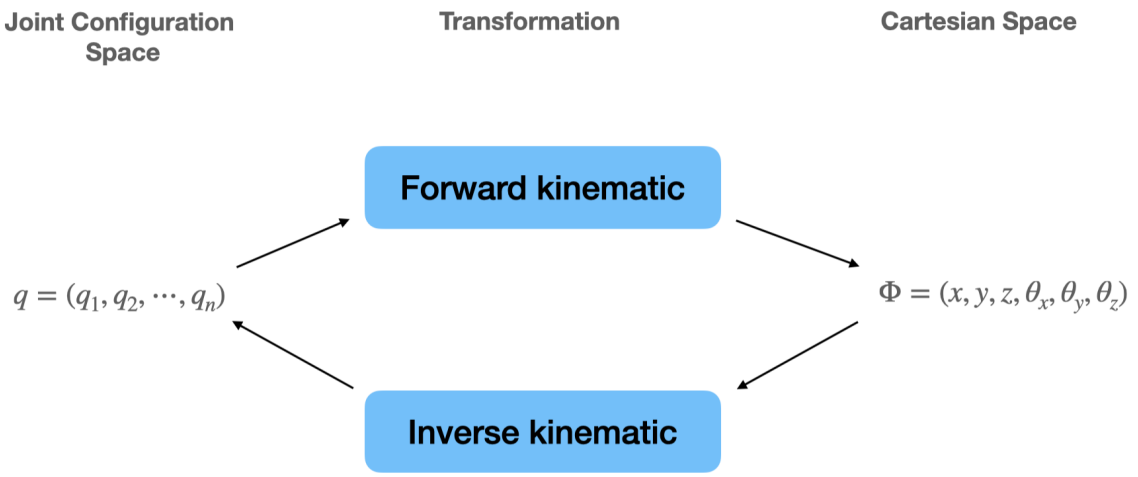


Figure 2.4: The relationship between forward kinematic and inverse kinematic

By using the Jacobian matrix, we can know how to solve the inverse kinematic problem numerically in an efficient way. The inverse kinematics is the problem of finding the joint configuration given a desired Cartesian end-effector configuration. We describe here only the numerical method for solving the inverse kinematic, and leave readers to consult the analytic method.

Suppose we express the end-effector frame using a 4×4 homogeneous transformation

$$T_e = \begin{bmatrix} R_e & p_e \\ 0 & 1 \end{bmatrix} \tag{2.15}$$

Here, the homogeneous transformation T_e represents the desired position and orientation of the end-effector. The goal of inverse kinematics is to find a solution or possible multiple solutions of joint coordinates $q_d = (q_{d,1}, q_{d,2}, \cdots, q_{d_n})$ such the equation $T_n^0(q_d) = T_e$.

Now, let $\Phi_d \in \mathbb{R}^m$ be a vector of Cartesian coordinate representation of homogeneous transformation T_E. For example, Φ_d could represent the center position of the wrist (m=3) or the position and orientation of the end-effector using a minimal representation of the orientation matrix (m=6). Assume that the forward kinematic function $f : \mathbb{R}^n \to \mathbb{R}^m$ is differentiable. Then, the goal is the find joint coordinates q_d such that

$$\Phi_d - f(q) = 0 \tag{2.16}$$

Given an initial guess q^0 , the kinematics can be expressed as the Taylor first order expansion

$$\Phi_d = f(q_d) = f(q^0) + \underbrace{\frac{\partial f}{\partial q}\Big|_{q_0}}_{J(q^0)} \underbrace{(q_d - q^0)}_{\Delta q} \tag{2.17}$$

Note that the Jacobian in Equation 2.17 is not the same Jacobian as presented in Equation 2.14. Here, we are take the direct partial derivative of the end-effector configuration, which depends on the parameterization of the end-effector orientation. In general case, $J(q^0)$ is not square $m \neq n$, we need to use the pseudo-inverse of the Jacobian to obtain

$$\Delta q = J_a^{\dagger}(q^0)(\Phi_d - f(q^0)) \tag{2.18}$$

To find a solution for Φ_d, we begin with an initial guess, q^0, and form a sequence of successive estimates, $q^1, q^2, q^3, \cdots$, as

$$q^k = q^{k-1} + \alpha_k J_a^{\dagger}(q^0)(\Phi_d - f(q^{k-1})) \tag{2.19}$$

Where α is the step size, always positive, which can be adjusted to aid convergence.

In this thesis, both forward kinematics and inverse kinematics are very useful for motion generation. For example, the position of the goal is given in the Cartesian space, if we control the robot in joint space, we first need to compute the corresponding joint space position for the goal. Therefore, if we control the robot directly in Cartesian space, we must compute the Cartesian error between the goal and the end-effector by using forward kinematics. In addition, the Jacobian matrix is a key component for linearizing the kinematics constraints of the robot in Chapter 3.

2.2 Motion Planning

2.2.1 Joint space versus Cartesian workspace

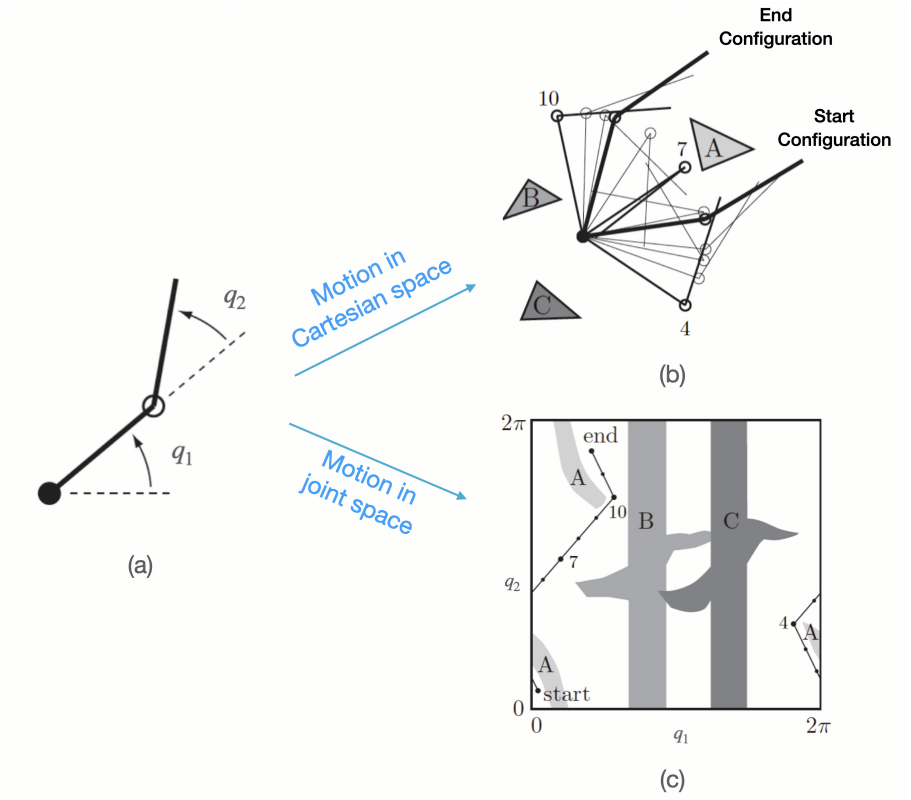


Figure 2.5: (a) a 2R robot manipulator with joints q_1 and q_2. (b) The manipulator is moving in workspace surrounding by obstacles A, B and C. (c) The same motion in the joint configuration space, the 2R arm's configuration is reduced to a point.

Motion planning is the problem of finding a robot motion from a start state to a goal state that avoids obstacles in the environment and satisfies other constraints such as joint limits or time optimality. Motion planning can be performed either in the joint configuration space (joint space)

or in the Cartesian workspace.

In Figure 2.5(a), the geometry of a manipulator with two degree of freedom (Dof) is presented, the configuration of the links is characterized by joint variables q_1 and q_2. The workspace and joint configuration path planning is shown in Figure 2.5(b) and Figure 2.5(c). Trajectories in joint space are computationally simpler for a robot manipulator because we don't need to compute the inverse kinematic problem online at each iteration. In addition, applying constraints in joint position, velocity and acceleration will be straightforward because the control input is directly in the joint space. However, in the presence of obstacles, trajectories in Cartesian workspace provide a straightforward visualization and interpretation of the robot's motion. Representing obstacle configurations in joint space remains a difficult task, as shown in Figure 2.5(c).

In this thesis, we choose to define trajectories in joint space because this allows simple application of constraints on joint range, velocity and acceleration. In addition, with the Jacobian matrix, the locations of the obstacles can be expressed as a function of the positions of the joints, this is explained in detail in Chapter 3. After introducing the two different spaces (joint space and cartesian workspace), we wanted to point out the difference between path planning (which only describes how to move from one configuration to another configuration) and trajectory planning (which imposes additional constraints on the path such as the duration of the motion, dynamics constraints and control input constraints).

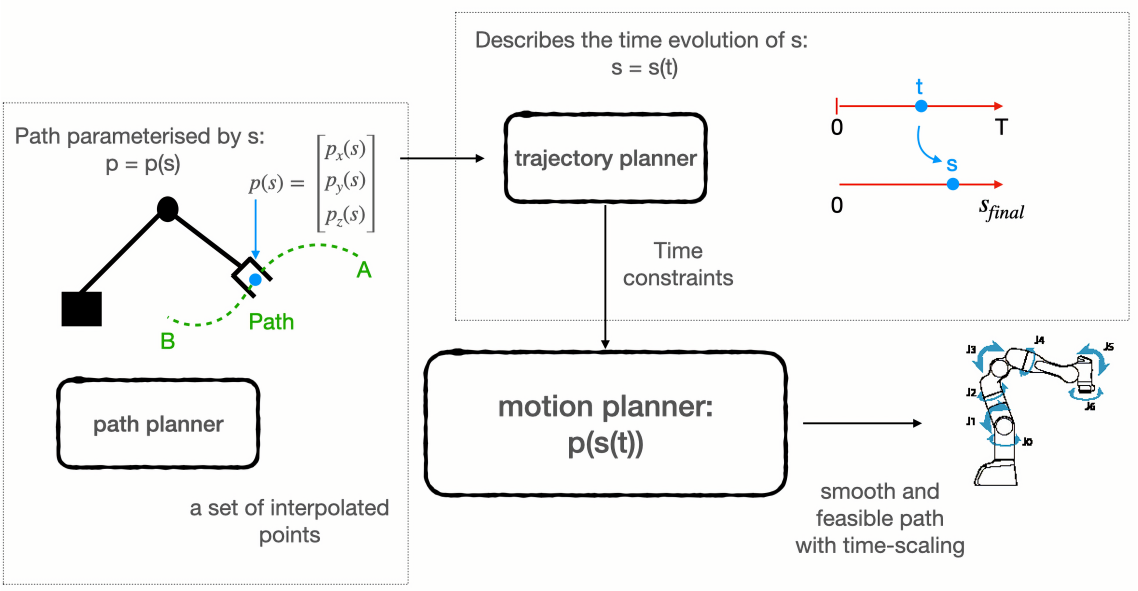


Figure 2.6: General motion planning = path planning + trajectory planning

The path planning problem is a purely geometric problem and is a subproblem of the general motion planning problem. It is a geometric description of the robot motion with the goal to find a collision-free path $p(s)$, $s \in [0,1]$ from a start configuration $p(0) = A$ to an end configuration $q(1) = B$ without considering the dynamics of the robot, the duration of the motion, the constraints on the motion and on the robot are shown on the left side of the Figure 2.6. Where $p(s) \in \mathbb{R}^n$ is the Cartesian position of the end-effector.

Once we have a set of interpolated points from the path planner defined by the parameter s, the trajectory planner will generate the time-scaling constraint on the curvilinear abscissa to obtain s(t). The curvilinear abscissa s(t) is defined by the path and the robot constraints such as velocity, acceleration and control input bounds, as shown in the top of Figure 2.6. In this thesis, there is no distinction between the path and the time law, since we determine a set of joint's acceleration that will cause the robot to satisfy the physical constraints (e.g. position,

velocity and acceleration constraints) and environment constraints (e.g. obstacle avoidance) with respect to some performance criteria, futhermore, we can use Euler integration applied to the joint's acceleration to obtain the joint space trajectory.

2.2.2 Local versus global planning

The problem of path planning is divided into local and global planning [LaValle2006]. In global path planning the robot has a complete knowledge of its environment and aims to find the shortest collision-free path for a complex workspace respecting all the constraints of the robot. In local path planning, the robot has little or no information about its environment and must instead take has to reactive approach to planning its path based on its perception of the environment.

Global path planning methods such as roadmap [Amato1996], [Hsu2006], cell decomposition [Lingelbach2004] and potential field [Khatib1986], [Hwang1992] have been proposed. Roadmap is defined as an union of curves such that for all start and goal points in free configuration space (robot configuration without collision with obstacles) that can be connected by a path [Choset2005]. The roadmap method is based on a graph-searching algorithm in the free configuration space, the well-known types of roadmaps are the visibility graph [Huang2004] and the Voronoi diagram [Bhattacharya2008]. The cell decomposition method divides the robot configuration space into cells. After this decomposition a connectivity graph between the initial configuration and the final configuration is constructed according to the adjacency relationships between the cells. The main disadvantage of the roadmap and cell decomposition methods is their high computational requirements, and in the case of dynamic obstacles, a replanning would be required. The potential field method considers the robot as a particle moving under the influence of a potential field that is determined by the set of obstacles and desired goals. Obstacles are assigned as repulsive forces while the goal position are assigned as the attractive force. However, this method is only effective in a stationary environment with stationary obstacles and suffers from the existence of a local minimum where the robot gets stuck. Thus, global planning is not a suitable solution for real-time planning of a robot manipulator in a dynamic environment.

On the contrary, local path planning is much computational efficient and do not need a complete information about the environment. The path planner will react to obstacles and generate collision free path in real time. Thus, this method are well suitable for real time application and for dynamic environment that requires replanning. However, local planning do not guarantee finding a solution and can stop in a local minimum even if a collision free path exist.

In this thesis, we propose a reactive trajectory planning algorithm that works in real time avoiding dynamic person. Chapter 3 presents the general framework for local motion generation based on a model predictive control (MPC) approach. Chapter 4 presents method to detect and predict the motion of the dynamic person to anticaipate the future and help collision avoidance by reacting to it in advance.

2.3 Optimisation Problem

The motion planning problem can be expressed as an optimal control problem where some performance criteria must be optimized while respecting given constraints. The general problem can be written in the following [Lynch2017]:

$$\underset{u(t),q(t),T}{\text{minimize}} \; J(u(t),q(t),T) \tag{2.20a}$$

$$\text{subject to } \forall t \in [0,T], \dot{x} = f(x(t),q(t)), \tag{2.20b}$$

$$\forall t \in [0,T], u \in \mathscr{U}, \tag{2.20c}$$

$$\forall t \in [0,T], q(t) \in \mathscr{C}_{free}, \tag{2.20d}$$

$$x(0) = x_{start}, \tag{2.20e}$$

$$x(T) = x_{end}, \tag{2.20f}$$

For the sake of simplicity and without loss of generality, the variables here are all one dimension. The variables u(t), q(t), T are the parameters to be optimized, corresponding to the robot's control input, the joint configuration and the total trajectory time, respectively. $J(u,q,T)$ (Equation 2.20a) is the perfomance criterion, also called the cost function and Equation 2.20b represents the robot dynamics. The admissible control input space and the admissible joint configuration space (without collision with obstacles) are denoted by $\mathscr{U}$ and $\mathscr{C}_{free}$, respectively. The initial and final state constraints are given by Equation 2.20e and Equation 2.20f. Such a general form can be very hard to solve if the cost function and constraints are nonlinear. It may be very difficult to find a feasible point solution (i.e., the tuple (u,q,T) that satisfies all equality and inequality constraints) and suffer from local optimum problem.

It turns out that if the objective function is convex and all constraint functions are affine, then, any local minimum is also the global minimum and the feasibility of the convex problem can be determined without ambiguity. The well-known convex optimization problem is called a quadratic program (QP) if the objective function is quadratic and all constraint functions are affine. A quadratic program is expressed in the form:

$$\underset{u,q,T}{\text{minimize}} \; \frac{1}{2}\begin{bmatrix} u(t) & q(t) & T \end{bmatrix} H \begin{bmatrix} u(t) \\ q(t) \\ T \end{bmatrix} + g^T \begin{bmatrix} u(t) \\ q(t) \\ T \end{bmatrix} \tag{2.21a}$$

$$\text{subject to } G \begin{bmatrix} u(t) \\ q(t) \\ T \end{bmatrix} \leq b \tag{2.21b}$$

$$C \begin{bmatrix} u(t) \\ q(t) \\ T \end{bmatrix} = d \tag{2.21c}$$

In the quadratic form (Equation 2.21), H is the positive definite symmetric matrix and g is the gradient of the objective function. G is a real matrix and b a vector denote with dimension equal to the number of inequality constraints. By taking $H = 0$, the quadratic form becomes another well-known convex optimization problem (linear programs). Several approaches are proposed to solve convex optimization: Interior Point [Wright1997], Simplex Algorithm [Murty1988]. In a convex optimization problem, if there is an optimal solution, it will be the global solution [Boyd2004a].

2.4 Model predictive control

Model Predictive Control (MPC) is an advanced method that is used to control a system while satisfying a set of constraints. It has been widely used in the 1980s in various industrial sectors, especially in chemical plants and oil fields. In MPC, the model is used to predict the future behavior of the system over a finite time window, the horizon. Based on these predictions and the current measured/estimated state of the system, the optimal control inputs with respect to a

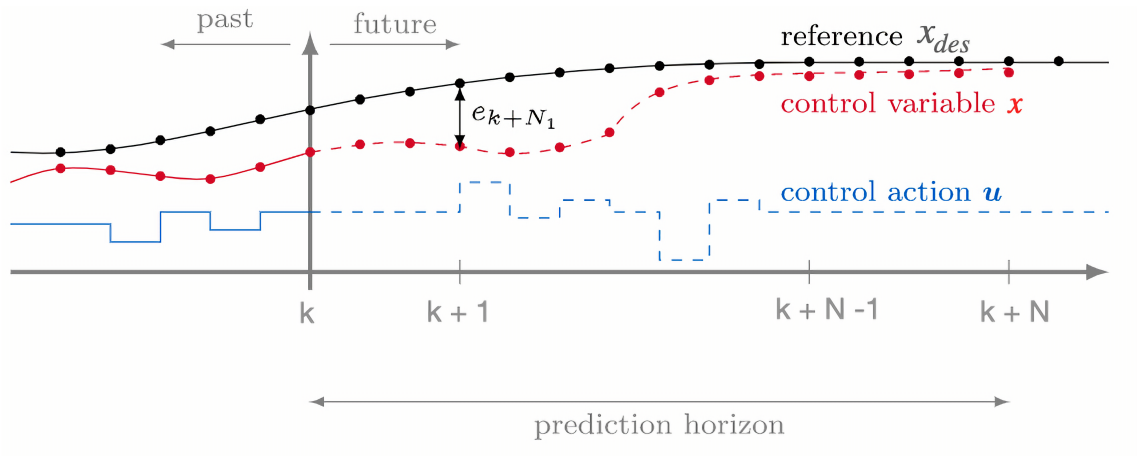


Figure 2.7: Figure illustrating how model-based predictive control works with horizons N

defined control objective and subject to system constraints are computed as shown in Figure 2.7. After a certain time interval, the measurement, estimation and computation process is repeated with a shifted horizon. For this reason, this method is also called receding horizon control (RHC). As we presented in the previous section, the motion planning problem can be expressed as an optimal control problem, and model predictive control can also be expressed as an optimization problem. Finally, methods developed to solve optimization problems can be efficiently applied to solve the optimal control input for MPC.

In Chapter 3, we present our online collision-free motion planner in the MPC framework, which not only improves the reactivity of the system, but also presents more accurate local linear approximations of the collision avoidance constraints, and also provides passive safety by an appropriate choice of terminal constraint.

2.5 Deep Learning for vision systems

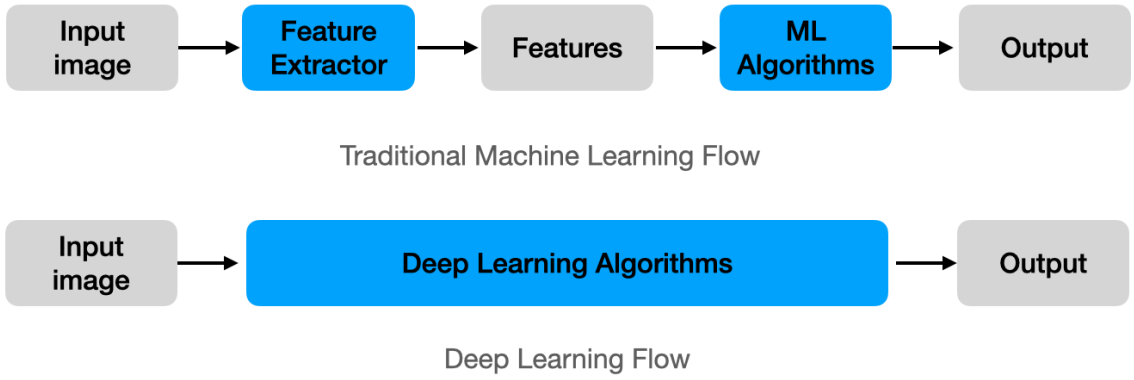


Figure 2.8: Illustration of traditional machine learning and deep learning working flow

In Section 1.3, we have shown that an autonomous robotic system is based on a perception-decision-action architecture. The perception module is thus a key component for the motion plan-

ner to generate a collision-free trajectory in a dynamic environment such as a human-robot collaboration scenario. Traditional computer vision algorithms use hand-crafted feature descriptors (e.g. HOG [Shu2011], SIFT [Lowe2004], etc) along with statistical machine learning (ML) classifiers to detect and track objects, as shown at the top of Figure 2.8. The difficulty with this traditional approach is that it is necessary to choose which features are important in each given image. It's up to the author's judgment and a long process of trial and error to decide which features best describe different classes of objects.

In fact, Deep Learning (DL) introduces the concept of end-to-end learning where the algorithm is just given a dataset of images which have been annotated with true labels. Then, the DL model is trained on the dataset to discover the underlying patterns and learn to predict the corresponding output, as shown below in Figure 2.8. Nowadays, DL has become an active research area in the robotics community [Pierson2017] and [Caldera2018]. In the human-robot collaboration scenario, DL has been applied for hand detection and localization [Gao2020], human motion recognization [Wang2018], human motion prediction [Liu2019], etc.

In this work, we apply DL models to the image coming from an RGB-D depth camera sensor to localize the human skeleton and thus predict its motion over a short time horizon. The predicted information is then incorporated into the MPC framework to generate a collision-free trajectory. In the following, we present some key concepts of DL to explain how the algorithms work.

2.5.1 Neural Networks

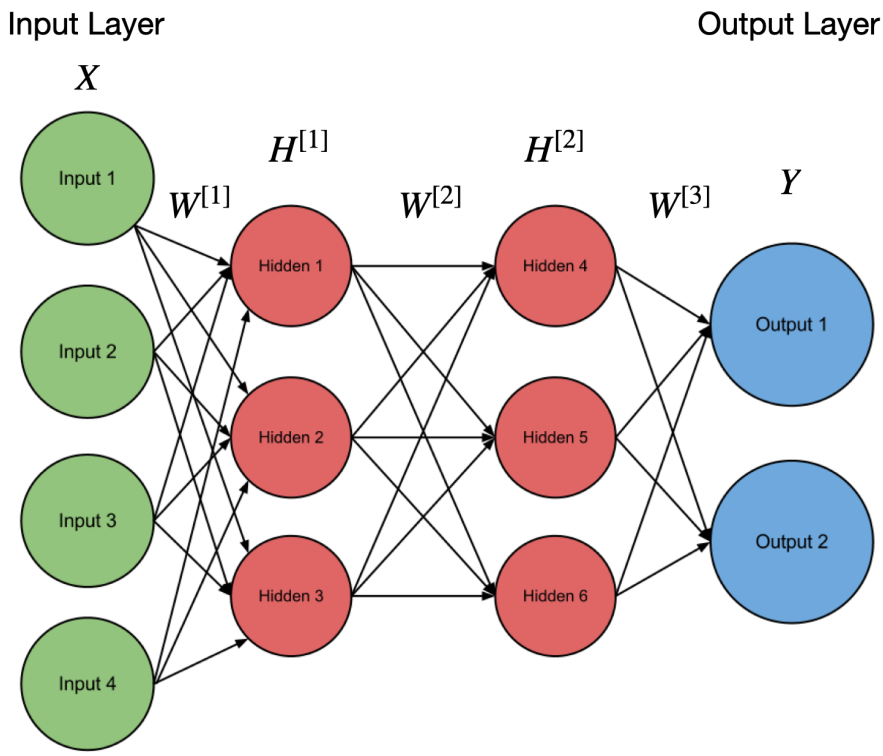


Figure 2.9: A deep network with multiple hidden layers, which transform the input representation to the output layer.

The foundation of deep learning is based on a multi-layer neural network, as shown in Figure 2.9. Each layer consists of a number of neurons that have weighted connections to each neuron in the previous layer. The neuron then forms an output by passing the weighted sum of its inputs

through some nonlinear activation function (for example a relu activation, $relu(a) = max(0,a)$). This hidden layer output is sent to the next layer. Ideally, each layer of features will represent a better abstraction of the input data, so that the final layer of features will be a better representation for some classifier than the raw features fed into the network. The mathematical representation for this simple two-layer deep network is:

$$h_j^{[1]} = \sigma(\sum_i^n o_i W_{i,j}^{[1]}) \tag{2.22}$$

$$h_j^{[2]} = \sigma(\sum_i^{M_1} h_i^{[1]} W_{i,j}^{[1]}) \tag{2.23}$$

$$y_j = \sigma(\sum_i^n h_j^{[2]} W_{i,j}^{[1]}) \tag{2.24}$$

Where $W^{[l]}$ are the network weights for layer l, $h^{[l]}$ as the corresponding hidden output, M_l as the size of this hidden representation of layer l, o as the raw input features with dimension n and y_j as j-th predicted output of the network. Not much can be done with these simple two-layer networks, but the structure of deep learning is very flexible and can be extended or modified to make it deeper and more complex. This modularity is the power of deep learning.

2.5.2 Convolutional Neural Networks

The input layer as shown in Figure 2.9 is a vector with n-dimensional features, but images are 3D tensors, with a height, width, and number of channels to represent colors. The spatial property of an image is not be preserved when it is applied to a neural network with the structure as shown in Figure 2.9. A convolutional neural network (ConvNet/CNN) is a deep learning architecture that is very efficient at handling structured input data such as an image.

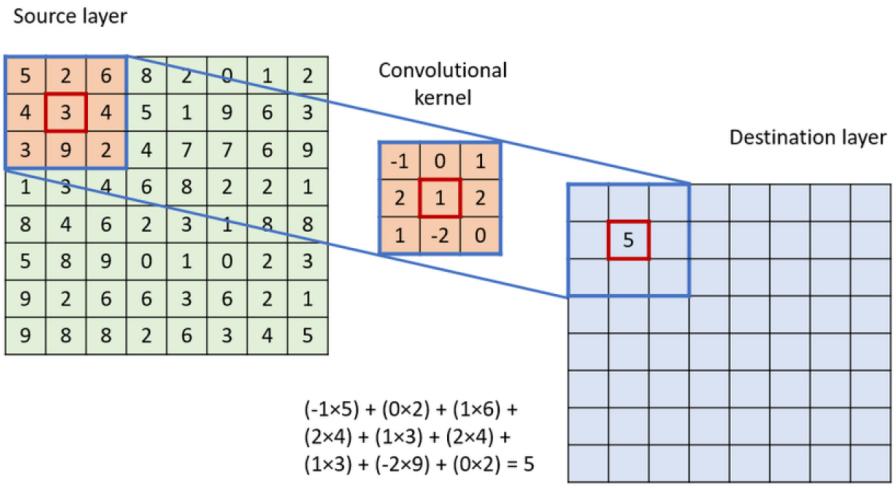


Figure 2.10: Illustration of convolution operation applied to an input layer.

The name Convolutional Neural Network means that the network uses a mathematical operation called convolution instead of general multiplication in some of its layers. The convolution operation is as follows:

$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(i-m, j-n) K(m,n) \tag{2.25}$$

Where I is the input two-dimensional grayscale image as input and K is also a two-dimensional matrix which is called the kernel. The output S is called as the feature map. The convolutional operation is illustrated in Figure 2.10. The kernel is usually smaller than the input which can detect small and meaningful features such as edges that occupy only a few hundred of pixels. This allow us to store fewer parameters in the kernel, which both reduces the model's memory requirements and improves its statistical efficiency. This idea is called to as sparse weights.

In the traditional neural network described above, each element of the weight matrix is used exactly once to compute the output of a layer. In a convolutional neural network, however, each element of the kernel is used at each position of the input. Again, this idea of parameter sharing greatly reduce the memory required to store the kernel weight. As shown in Figure 2.10, the same convolutional kernel is applied to each location of the input layer. To get an idea of the explosion of parameters in traditional neutral networks, if we take as an example a colored image input of 64×64, the number of weights on just a single neuron of the first layer increases to 12,288 $(64 \times 64 \times 3)$ plus one for the bias parameter.

Another important feature in convolutional neural networks is the pooling layer, which will simply performs downsampling along the spatial dimensionality of the given input, thus further reducing the number of parameters and the computational complexity of the model.

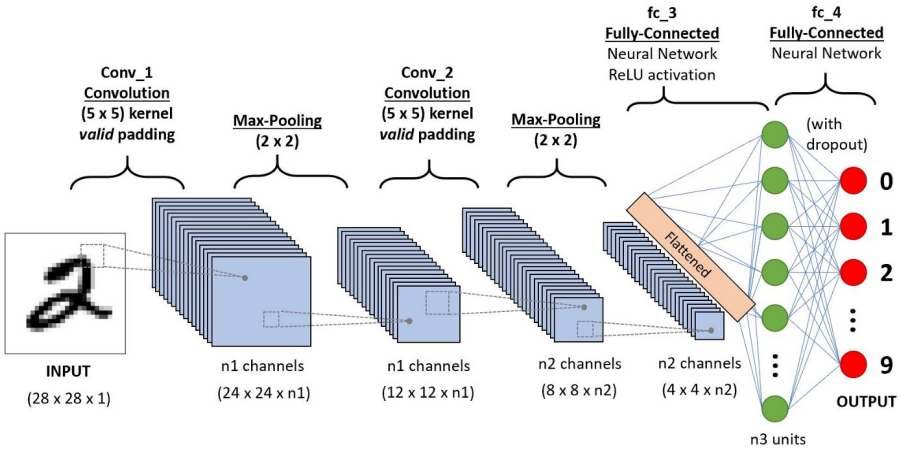


Figure 2.11: A common form of convolutional neural network architecture in which we successively apply convolution, relu nonlinear activation and pooling before going fully connected layer.

In Figure 2.11, we present the most common basic form of a convolutional neural network architecture to handwritten digit recognization task. We successively apply convolution operation (use kernel to extract interesting features) followed by a nonlinear activation such as relu. Then, we apply max-pooling to the output feature maps from the convolutional-relu layer to further reduce it's size. Then, a final fully-connected layer is used to make the prediction, in this case to predict the digit between 0 and 9.

2.5.3 Recurrent Neural Network

We discuss above that the convolutional neural networks are very efficient to handle hierarchical structure such as an image, the CNN architecture only looked at one image and then output a single prediction. However, remaining that in our human-robot collaboration scenario, we want to predict the future motion of the person during a time horizon. Therefore, we need a more appropriate network architecture to handle a sequence of images or data and produce a sequential prediction. A more formal definition is given an observation sequence $o = \{o_1, o_2, \ldots, o_{T_1}\}$ and its

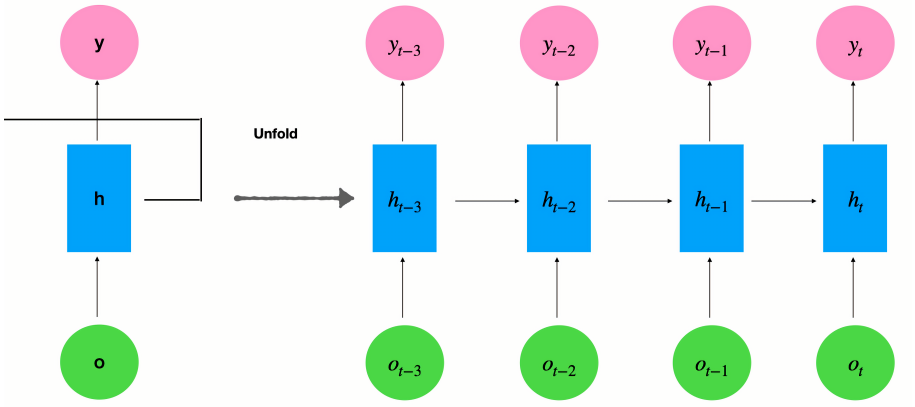


Figure 2.12

corresponding label $y = \{y_1, y_2, \ldots, y_{T_2}\}$, we want to learn a network architecture $f : o \rightarrow y$. Where T_1 and T_2 are the length of the sequences.

Recurrent Neural Networks (RNN) are a variant of deep learning models that capture sequential data via recurrent connections. It models a dynamic system where the hidden state h_t depends not only on the current observation o_t, but also relies on the previous hidden state h_{t-1}. A simple single-layer RNN is shown in Figure 2.12, and the corresponding mathematical notation is as follows:

$$\begin{aligned} h_t &= \sigma(W_{xh} o_t + W_{hh} h_{t-1} + b) \\ y_t &= W_{hy} h_t \end{aligned} \tag{2.26}$$

Where o is the input sequence containing a sequence of observations. To keep things simple we assume that each observation is a scalar value with single feature. The idea can be extended to a n-dimensional feature vector. y is the output sequence of the network, h is the vector storing the values of the hidden units or state, which can contain a number m of hidden units. W_{xh} are weights associated with inputs in the recurrent layer, W_{hh} are weights associated with hidden units in the recurrent layer, W_{hy} are weights associated with hidden to output units, and b is the bias in recurrent layer. the nonlinear function σ corresponds to the activation function and is usually taken as Tanh function ($\frac{e^x - e^{-x}}{e^x + e^{-x}}$) or relu function. However, the standard RNN models have two major problems, exploding gradients and vanishing gradients. Exploding gradients occur when the algorithm places a high values to the weights. Vanishing gradients occur when the values of a gradient are too small and the model stops learning or takes way too long as a result. In the recent days, the state-of-the-art variations of RNN models are Long Short-Term Memory (LSTM) and Gate Recurrent Unit (GRU). A detailed description of LSTM and GRU can be found in [Staudemeyer2019]. In this thesis, we don't differentiate between RNN models and their variations, because they share the same core idea and only differ in their architecture.

In this thesis, we will use convolutional neural network for person detection in an image, and we will also extract it's key-points. Moreover, we need to predict the person's motion to enable our model predictive control framework to actively compute a collision free trajectory.

2.6 Conclusion

In this chapter, we have reviewed several techniques in robotics and in computer vision that have some common ground with the research work considered in this thesis. Our we goal here

is to present the current challenging requirements considered in this thesis, and pointing out the contribution of the presented work in this regard.

The goal of this thesis is to provide an online trajectory generator for robotic manipulators in a dynamic environment such as moving people nearby. Specifically, our goal is to design a reactive trajectory generator satisfies the following requirements: 1) Completes task as efficiently as possible under given constraints. 2) Generates collision-free trajectories. 3) Works in real-time. To help our online trajectory generator to perceive the dynamic environment, we apply current state-of-the-shelf computer vision algorithm such as Convolutional Neural Network. CNN is very effective to handle image data, therefore, we use it to effectively detect the location and the pose of a moving person around the robot. In addition, the ability to incorporate the prediction of the person's motion provides important advantage for our online trajectory generator. The online trajectory generator will be more reactive to avoid potential collisions because it predicts the future changes. Thus, the motion prediction is handled by the recurrent neural network.

3

Online motion generation in a shared workspace

Contents

Path planning in the presence of a dynamic person is a challenging problem due to the added complexity. To account for the dynamic nature of the person, the robot must predict the future trajectories of these obstacles in order to plan its own path accordingly. As a result, the trajectories must be actively re-plannaed in order to handle the incorrect person dynamics and inaccurate sensor data.

In this chapter, we will introduce the real-time trajectory generator under the model predictive control framework described in the end of section 1. First, we present the state-of-the-art in robot collision avoidance. We compare different methods and summarize their advantages and disadvantages on the Tab.3.1. Based on the comparison, we argue that formulating the planning problem as an optimization problem is the most appropriate method for human-robot collaboration. Then, we give the mathematical definition of the trajectory generation problem in the dynamic environment and integrate it into our MPC framework. Finally, we illustrate the approach with several simulations with different settings.

3.1 State of the art

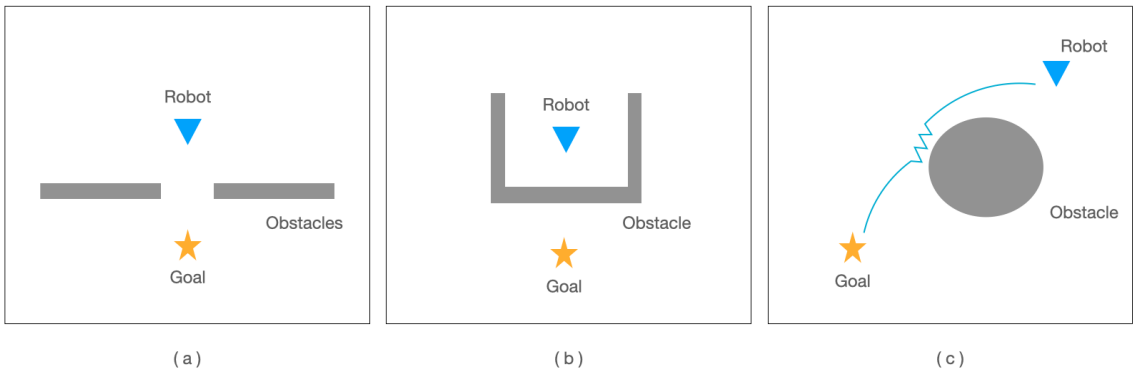


Figure 3.1: Illustration of local minima in a potential field: (a) small distance between obstacles, (b) the robot is surrounded by obstacles and the goal is opposite the exit, (c) the goal is too close to the obstacle.

New, safer manipulator robots can share their workspace with humans, thanks to advanced mechanical and control design that make sure that potential collisions between the robot and humans result in a low (5%) probability of serious injury even at speeds as high as 2 $m.s^{-1}$ [Albu-Schäffer2007]. Collisions would better be avoided nevertheless, not only because of the remaining risk, however low it is, but also because each collision disrupts the tasks of both the robot and the human.

The most popular and well known approach to dynamic environment planning for robots is the use of potential fields [Khatib1986], [Connolly1990], where repulsive forces push the robot away from obstacles and an attractive force pulls the robot towards the goal. In order to be fast and thus limit the computation time, this reactive scheme consists of computing at each time step only the next action that will move the robot closer to the goal without colliding with the environment. However, this method can cause the robot to get stuck in local minima and suffer from oscillations, as shown in Fig. 3.1. The oscillation problem of the potential field (Figure 3.1(c)) can be reduced using Newton's method for calculating the gradient of the potential [Ren2006]. The local minima problem occurs when all attractive and repulsive forces cancel each other out,

this happens when the exit is a small distance between obstacles or the goal is very close to the obstacles (Figure 3.1(a)) or when the robot is surrounded by obstacles and the exit is opposite to the goal (Figure 3.1(b)). The local minima can be avoided by combining global and local information as shown in the elastic band framework [Quinlan1993], adding a virtual obstacle [Lee2003] around local minima to help the robot to escape from it, using harmonic potential [Kim1992] functions to construct a local minima free potential field by finding a solution of the Laplace equation. However, the drawback lies harmonic functions are computationally expensive and difficult to implement in a dynamic environment that changes with time and some implement in a dynamic environment do not ensure the existence of trajectory [Garrido2010].

Figure 3.2: Human-robot collision avoidance in depth space with potential field method.

An example of robot collaboration using artificial potential fields is shown in Figure 3.2, [Flacco2012b] uses a depth camera to estimate the distance between a robot and an obstacle (including a human), which is used to generate the repulsive forces to control the robot for generic motion task. [Polverini2014] introduces a kinetostatic safety field to estimate the level of danger near rigid bodies. The safety field includes the relative position and velocity of the robot and the human, but is also influenced by the real shape and size of the human. A control input is then derived from the cumulative safety field. In the above method, the obstacle is surrounded by an envelope that is larger than its real dimensions, which may unnecessarily limit the robot's movement.

Rather than generating a motion in the workspace, configuration space planning provides a unified aspect of the trajectory generation and control problem, and it can also reduce the complexity of representing the robot topology in the workspace. The configuration of a robot is a specification of the position of each point on a robot, and the configuration space is the set of all possible robot configurations. Thus, the planning problem in workspace as described by the potential field method can be reduced to path planning for a single point in the configuration space (Section. 2.2). However, the difficulty of planning in the C-space lies in the obstacle representation, especially for robots with many DOFs and for dynamic obstacles. Two well-known approaches have been developed to deal with this drawback, sampling-based algorithms such as rapidly-exploring random trees (RRT) in [Kuffner2000] and probabilistic roadmaps (PRM) in [Kavraki1996]. By randomly sampling in the C-space, a tree or a roadmap can be constructed to determine a connection between the start and the end configuration in C-space. The validation of the samples for collision avoidance is checked by mapping the sample into the workspace using forward kinematics. An efficient algorithm such as A^* [Duchoň2014] can be applied to the tree or the roadmap to find the path. We have summarized the advantages and disadvantages of the above described motion

planners in Tab. 3.1.

method	Advantages	Disadvantages	Suitability for Human-Robot collaboration
Potential field	Online capability. No need for obstacle representation in C-space.	Need to escape the local minimum. No optimal consideration of the motion. Complex kinematics and dynamics representation.	Not suitable.
Planning in C-space	The topology of the robot is reduced to a point in C-space. No need to represent the kinematics.	Representation of obstacle topology in C-space is very time-consuming and complicated. Not online computable for high DOF robots and dynamic obstacles.	Not suitable due to the complicated obstacle representation.
Sampling-based planner	No need for kinematic representation as path is planned in C-space. No need for obstacle representation in C-space because collision checking is done in the workspace with forward kinematics. Real-time capability.	Heuristics in the approach. Finite number of samples is considered. Dynamics are rarely considered.	Not suitable. Difficult to formulate the heuristics in the optimization problem. Difficult to check collisions from the C-space to the workspace.
Optimisation formulation	No need for obstacle representation in C-space. Real-time capability. Easily integrated into the predictive control framework.	Kinematics or dynamics representation.	Suitable. Optimal solution with constraints satisfaction.

Tableau 3.1: Comparison of different planner approach

The above methods have important drawbacks that make them unsuitable for human-robot collaboration applications, sucha as the local minimum problem and the complexity of managing the dynamic environment in the potential field, the time-consuming transformation of the obstacle topology in the configuration space and the heuristic approach in the sampling-based planner. None of these methods easily incorporate constraints such as robot dynamic limits, motion duration into the formulation, and they do not consider the motion of obstacles in the future or in a time horizon. This has been pointed out by [Lasota2015] that simply preventing collisions as they are about to occur can lead to inefficient human-robot interaction and negatively impact perceived safety and comfort, therefore predicting the motion of obstacles and acting in advance is necessary to improve human-robot collaboration. To address all these drawbacks, we formulate our motion planning problem as an optimization problem, with equality and inequality constraints. In addition, it can be easily incorporated into a predictive framework such as model predictive control. In the following, we describe some optimization-based applications in human-robot collaboration and explain the problem formulation in more detail.

In [Meguenani2015, Joseph2018], they constrain the robot's kinetic energy by reducing the robot's velocity when an obstacle comes close, but the robot does not dynamically avoid obstacles. In [Balan2006], a collision-free trajectory is determined by solving an optimization problem that includes predictions of the robot and the worker motions. In the formulation of optimization problem, the robot and the worker are represented by spherical geometry and the collision checks are computed for a number of time-steps ahead based on the prediction which is used to formulate a cost function. In [Tsai2014, Ding2011b], the model predictive control framework is used to gen-

erate the collision-free trajectory online. They differ by the definition of constraints, in [Tsai2014] the momentum of the robot links are used to generate additional safety measures with the relative distance between the robot and the person. The total distance safety index plus the momentum safety index are used as constraints for obstacle avoidance. In [Ding2011b], a safety critical region in the workspace is determined by predicting the future positions of the human, and this region is considered as a constraint that the robot is forbidden to enter.

A drawback of the above prediction methods is the distance computation strategy, because they only compute the distance at one time step which cannot ensure the problem of interpenetration during the interval of two time steps. The classical approach to monitor the distance between a robot and its environment is with the Gilbert–Johnson–Keerthi distance algorithm, which provides a pair of closest points between two objects [Gilbert1988]. But monitoring only a pair of closest points can lead to catastrophic failures (collisions), since the closest points can change abruptly as objects move [Kanehiro2008].

In this work, we define the distance by establishing the existence of a separating plane between the robot and the obstacles. If such a plane exists, then we have evidence that the robot and the obstacles don't collide as illustrated in [Brossette2017]. In addition, this distance is immune to changes in the closest points. Furthermore, the separating plane provides continuous trajectory safety over entire time intervals at very low computational cost [Schulman2014]. The Figure 3.3

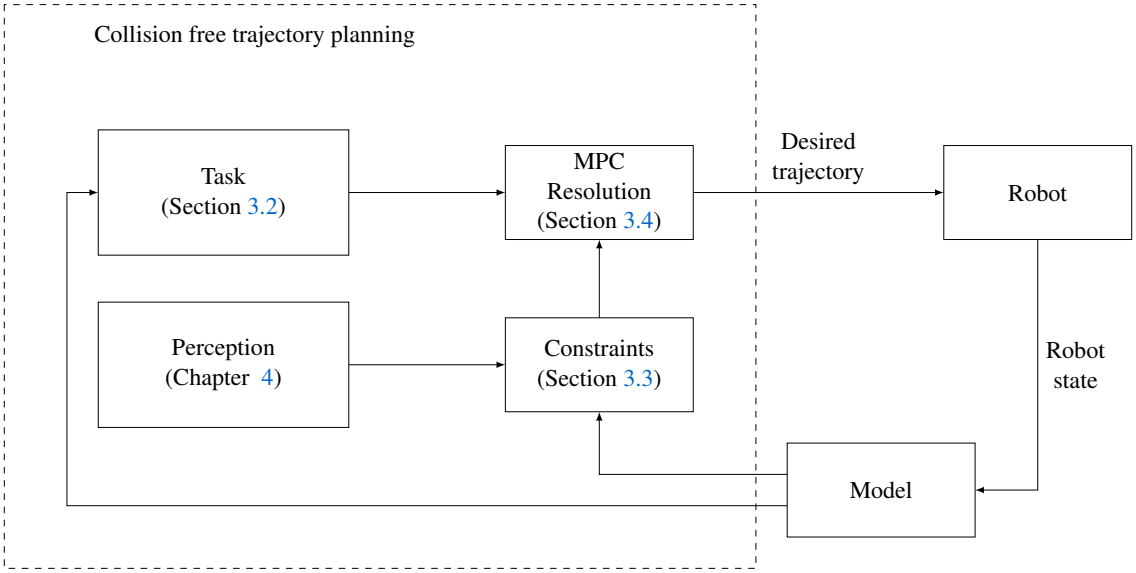


Figure 3.3: The structure of our predictive control framework

gives a global structure of the rest of the chapter and the goal of the following section is to present these aspects:

1. Define online collision motion generation as an optimization-based planning problem. Properly define the task (Section 3.2), physical constraints, and obstacle avoidance constraint (Section 3.3) in the optimization problem.

2. Define the collision avoidance constraint by separating plane (Section 3.3).

3. Extend the above method and incorporating passive safety into MPC framework (Section 3.4).

The quality of the generated trajectory under Model Predictive Control (MPC) depends in particular on the prediction of the dynamic changes in the environment (e.g. the future positions of the person). Concerning the prediction of human motion, we will give a detailed discussion in Chapter 4. In this chapter, we assume that the prediction of human motion is available, so we

concentrate on the problem of formulating and solving the online collision-free motion generation under the MPC framework.

3.2 Task definition

Consider the motion of an n link serial manipulator with joint position $q \in \mathbb{R}^n$ and piecewise constant acceleration over time periods Δt, represented as a linear discrete time system

$$x_{k+1} = Ax_k + Bu_k \tag{3.1}$$

where

$$x_k = \begin{bmatrix} q_k \\ \dot{q}_k \end{bmatrix} \in \mathbb{R}^{2n}, \tag{3.2}$$

$$A = \begin{bmatrix} I & I\Delta t \\ 0 & I \end{bmatrix} \in \mathbb{R}^{2n \times 2n}, \tag{3.3}$$

$$B = \begin{bmatrix} I\frac{\Delta t^2}{2} \\ I\Delta t \end{bmatrix} \in \mathbb{R}^{2n \times n} \tag{3.4}$$

with I an n$\times$ n identity matrix and $u_k = \ddot{q}_k$ the piecewise constant acceleration.

Using the above discrete system (Equation 3.1), the objective function formulated in joint-space is defined as the error to be minimized between the desired state x^{des} and the current joint state x_k with the desired acceleration u_k^{des}:

$$\underset{u_k}{\text{minimize}} \quad ||x_{k+1}^{des} - x_{k+1}||_Q^2 + ||u_k||_R^2 \tag{3.5}$$

Where Q and R are symmetric positive semie-definite weighting matrices, the ratio R/Q allows to balance the minimization of the input acceleration while tracking the desired state x_{k+1}^{des}.

Formulating our objective as a quadratic form (Equation 3.5) means that the task can be written as functions of positive semie-definite matrices. Thus, it can be transformed into a quadratic program (Equation 2.21a), which is convex and admits a unique global solution [Boyd2004b].

3.3 Constraints definition

The robot is subject to various forms of constraints imposed by it's own mechanical system or by the environment. The equation of the linear discrete time system (Equation 3.1) is a form of equality constraint. The physical bounds on joint position, velocity and acceleration are a form of inequality constraint. And when the robot shares the workspace with an operator, the collision avoidance constraint is also expressed as an inequality constraint. By expressing all the above constraints as a function of the robot joint acceleration, it is possible to transform the constraint into a quadratic programming (QP) problem as defined by the Equations 2.21b - 2.21c.

3.3.1 Physicals constraints

Joint acceleration limits The joint accelerations are directly linked to the control variable. The corresponding constraint is simple:

$$\underline{u} \leq u_k \leq \overline{u} \tag{3.6}$$

$u_k = \ddot{q}_k$ is the piecewise constant acceleration defined in Equation 3.5. This is the control variable to be solved by the QP solver. $\underline{u}$ and $\overline{u} \in \mathbb{R}^n$ are acceleration limits imposed by the joint.

According to the quadratic programming problem form presented in Equation 2.21b, the acceleration constraint can be formulated as

$$\begin{aligned} lb_u \leq & G_u u_k \leq ub_u \\ \text{with } & G_u = I_n \\ & lb_u = \underline{u} \\ & ub_u = \overline{u} \end{aligned} \tag{3.7}$$

Joint velocity limits The joint velocity limits are defined as

$$\underline{\dot{q}} \leq \dot{q}_{k+1} \leq \overline{\dot{q}} \tag{3.8}$$

Where $\dot{q}_{k+1} \in \mathbb{R}^n$ is the robot joint velocity after control input u_k, $\underline{\dot{q}}$ and $\overline{\dot{q}} \in \mathbb{R}^n$ the joint velocity limits. Using Equation 3.1, we can directly select the joint velocity variable through a selection matrix $C_{dq} = \begin{bmatrix} 0_n & I_n \end{bmatrix}$. Thus, we can express $\dot{q}_{k+1}$ as a function of u_k.

$$\begin{aligned} \dot{q}_{k+1} &= C_{dq} x_{k+1} \\ &= C_{dq} A x_k + C_{dq} B u_k \end{aligned} \tag{3.9}$$

Equation 3.8 can be written in the form

$$\begin{aligned} lb_{dq} \leq & G_{dq} u_k \leq ub_{dq} \\ \text{with } & G_{dq} = C_{dq} B \\ & lb_{dq} = \underline{\dot{q}} - C_{dq} A x_k \\ & ub_{dq} = \overline{\dot{q}} - C_{dq} A x_k \end{aligned} \tag{3.10}$$

Joint position limits The joint position limits are defined as

$$\underline{q} \leq q_{k+1} \leq \overline{q} \tag{3.11}$$

Where $q_{k+1} \in \mathbb{R}^n$ is the robot joint position after control input u_k, $\underline{q}$ and $\overline{q} \in \mathbb{R}^n$ the joint position limits. Same as for joint velocity constraint, we define a selection matrix $C_q = \begin{bmatrix} I_n & 0_n \end{bmatrix}$ and then the joint position variable can be directly obtain from Equation 3.1. Thus, we can express q_{k+1} as a function of u_k

$$\begin{aligned} q_{k+1} &= C_q x_{k+1} \\ &= C_q A x_k + C_q B u_k \end{aligned} \tag{3.12}$$

Equation 3.11 can be written in the form

$$\begin{aligned} lb_q \leq & G_q u_k \leq ub_q \\ \text{with } & G_q = C_q B \\ & lb_q = \underline{\dot{q}} - C_q A x_k \\ & ub_q = \overline{\dot{q}} - C_q A x_k \end{aligned} \tag{3.13}$$

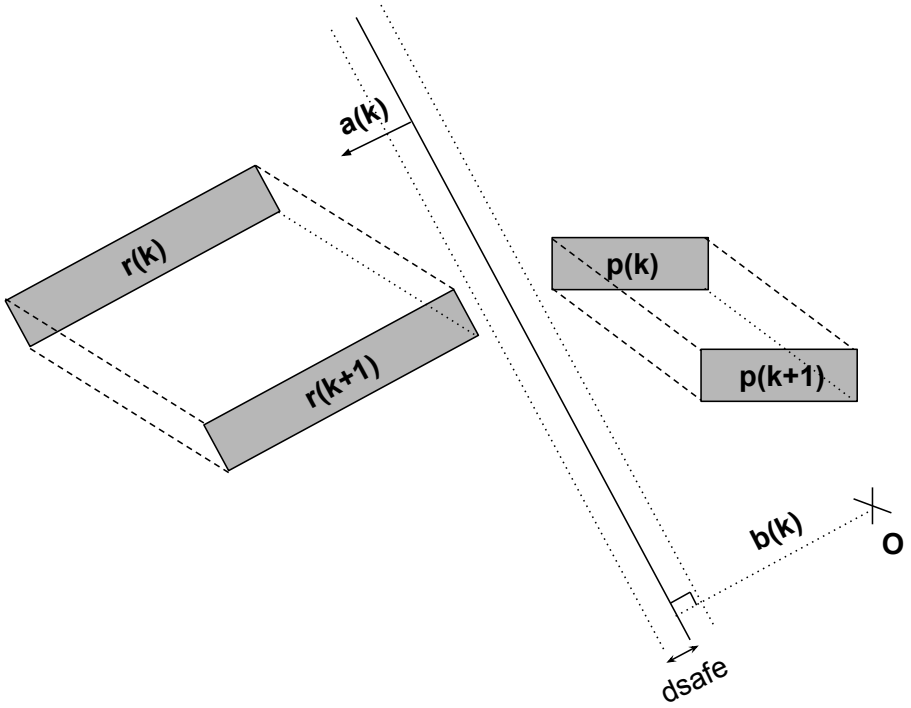


Figure 3.4: If there exists a plane such that all vertices r^i of the robot stay on one side of the plane and all vertices p^j of the person stay on the other side between instants k and $k+1$, we have evidence that they don't collide over this interval of time.

3.3.2 Collision avoidance constraint

The closest distance between the robot and obstacles is usually used to formulate collision avoidance constraints, but it is not very convenient. Because finding these closes points is difficult, and if an object is not strictly convex, the distance function of a pair of objects will be not be continuously differentiable. A different approach to closest distance is the idea of separating planes [Brossette2017], which can ensure the continuous collision avoidance between two intervals of time step.

The notion of a separating plane is illustrated in Figure 3.4). If we represent the different parts of the robot and the person as polyhedra with vertices $\{r^1, r^2, \cdots\}$ and $\{p^1, p^2, \cdots\}$ respectively. Thus, if there exists a plane defined by a normal vector $a_k \in \mathbb{R}^n$ and a scalar constant $b_k \in \mathbb{R}$ such that all vertices $p^j \in \mathbb{R}^3$ of the person remain on one side between instants k and k+1:

$$a_k^T p_k^j \leq b_k, \tag{3.14a}$$
$$a_k^T p_{k+1}^j \leq b_k, \tag{3.14b}$$

while all vertices $r^i \in \mathbb{R}^3$ of the robot stay on the other side, with some additional distance d:

$$a_k^T r_k^i \geq b_k + d, \tag{3.15a}$$
$$a_k^T r_{k+1}^i \geq b_k + d, \tag{3.15b}$$

then we have evidence that they don't collide over this interval of time [Schulman2014].

Optimisation formulation of separating planes

Given the robot trajectory computed at the previous sampling time and the current human motion prediction, we can find planes that maximize the distance d:

$$\underset{a_k, b_k, d}{\text{maximize}} \; d \tag{3.16}$$

subject to constraints (3.14) and (3.15). If the current human motion happens to be faster than the previously predicted one, it may not be possible to satisfy the constraints (3.15) with the previously computed robot trajectory and a positive distance d. In this case, we temporarily accept a negative "distance" and find the plane (a_k, b_k) that comes closest to separating the previously computed robot trajectory from the current human motion prediction. When this plane is used to compute a new collision-free robot trajectory in the next step, a fixed positive safety distance d_{safe} is again enforced. Note that the constraints (3.14) and (3.15) are linear with respect to a_k, b_k and d. For the maximization problem to be well-posed, we also need to bound the vector a_k to a unit norm ($\|a_k\| \leq 1$). This is a nonlinear constraint, but it can be efficiently approximated with linear constraints:

$$-\begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \leq a_k \leq \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}, \tag{3.17}$$

$$1 - \varepsilon \leq a_k^T \, a_k^p \leq 1 \tag{3.18}$$

where a_k^p is the vector a_k computed at the previous sampling time and ε is a small positive constant. In this way, the maximization (3.16) turns out to be a standard Linear Program. However, we found it preferable to smooth the variations of the separating planes by considering a regularized objective instead.

$$\underset{a_k, b_k, d}{\text{minimize}} \; -d + \alpha d^2 + \beta \|a_k - a_k^p\|^2 + \beta \|b_k - b_k^p\|^2 \tag{3.19}$$

with a_k^p, b_k^p the separating plane obtained at the previous sampling time and small weights α and β. This results in a linearly constrained QP that can be efficiently with off-the-shelf solvers.

3.4 MPC framework

3.4.1 Reformulate tracking problem without collision avoidance constraints

The idea of Model Predictive Control is to solve at each sampling time an optimal control problem over a finite horizon starting from the current state x_0. Consider a control sequence $\{u_0, u_1, \dots u_{N-1}\}$ of length N. A recursive application of the linear discrete time system (3.1) yields the resulting sequence of states $\{x_1, \dots x_N\}$:

$$\underbrace{\begin{bmatrix} x_1 \\ \vdots \\ x_N \end{bmatrix}}_{\mathbf{x}} = \underbrace{\begin{bmatrix} A \\ \vdots \\ A^N \end{bmatrix}}_{P_x} x_0 + \underbrace{\begin{bmatrix} B & 0 & 0 \\ \vdots & \ddots & 0 \\ A^{N-1}B & \cdots & B \end{bmatrix}}_{P_u} \underbrace{\begin{bmatrix} u_0 \\ \vdots \\ u_{N-1} \end{bmatrix}}_{\mathbf{u}}. \tag{3.20}$$

The objective of the MPC scheme is to

$$\underset{\mathbf{u}}{\text{minimize}} \quad \sum_{k=0}^{N-1} \|x_{k+1} - x_{k+1}^{des}\|_Q^2 + \|u_k - u_k^{des}\|_R^2 \tag{3.21a}$$

$$\text{subject to} \quad \forall k \in \{0,\dots N-1\},\ \underline{u} \leq u_k \leq \overline{u}, \tag{3.21b}$$

$$\forall k \in \{1,\dots N\},\ \underline{q} \leq q_k \leq \overline{q}, \tag{3.21c}$$

$$\forall k \in \{1,\dots N-1\},\ \underline{\dot{q}} \leq \dot{q}_k \leq \overline{\dot{q}}, \tag{3.21d}$$

$$\dot{q}_N = 0 \tag{3.21e}$$

where $\underline{q}$, $\overline{q}$, $\underline{\dot{q}}$, $\overline{\dot{q}}$, $\underline{u}$, $\overline{u}$ indicate minimum and maximum joint position, velocity and acceleration (we assume that $\underline{\dot{q}} \leq 0 \leq \overline{\dot{q}}$ and $\underline{u} \leq 0 \leq \overline{u}$). Here, the goal is to track a desired joint state trajectory x_k^{des} with acceleration u_k^{des}, so this optimal control problem takes the form of a simple linearly constrained Quadratic Program (QP) that can be solved efficiently with off-the-shelf solvers. A sample alternative is to consider a desired Cartesian motion of the robot's end effector using its forward kinematic model, but this makes the problem nonlinear. Finally, the robot follows the obtained optimal control sequence until the next sampling time, and a new optimal control problem is solved.

A key element of this approach is the terminal constraint (3.21e). It may seem to be an unnecessarily constraint on the robot's motion, but it provides recursive feasibility, guaranteeing that when the optimal control sequence is applied to the robot, it will always lead to new states of the robot where the optimal control problem (3.21) is once again feasible. This allows guaranteeing that all constraints in (3.21) will always be satisfied [Mayne2000], which is crucial. More precisely, this terminal constraint enforces that the robot is at rest at the end of the prediction horizon. When collision avoidance is introduced into this MPC framwork, this is what provides the passive motion safety guarantee, ensuring that the robot is able to stop and remain at rest before a collision occurs in the future. This is a central aspect of our proposed approach.

3.4.2 Reformulate tracking problem with collision avoidance constraints

We can compute a collision-free robot trajectory with the separating planes computed in the previous step. In the constraints (3.15), the positions of all vertices r^i depend on the kinematics of the robot, which is usually a nonlinear function of the joint position q. We propose to linearize the kinematics of the robot around the trajectory q_k^p computed at the previous sampling time:

$$\forall k \in \{1,\dots N\},\ r_k^i = r^i(q_k) \approx r^i(q_k^p) + J(q_k^p)(q_k - q_k^p) \tag{3.22}$$

where $J(q_k^p)$ is the Jacobian of the robot's kinematics. In this way, the constraints (3.15) are transformed into linear functions of q, and the MPC scheme (3.21) with these additional constraints is kept in the form of a linearly constrained QP, which can be solved efficiently with off-the-shelf solvers.

If we can't find a trajectory that satisfies these constraints because the current human motion has exceeded the robot's collision avoidance capabilities, we can continue with the trajectory obtained at the previous sampling time. It is designed to stop gracefully before a collision occurs, due to the terminal constraint (3.21e), at least with what could be predicted from the human motion. As long as our prediction is valid, this allows us to enforce passive motion safety. If our prediction happens to be invalid because the human made a completely unexpectedly fast motion, we can still rely on the intrinsic mechanical safety of the robot, with a low (5%) probability of serious injury even at speeds as high as 2 m.s^{-1} [Albu-Schäffer2007].

To summarize, for each time interval $[k, k+1]$, $k \in \{0,\dots N-1\}$, we compute separating

planes with QPs:

$$\min_{a_k,b_k,d_k} \; -d_k + \alpha d_k^2 + \beta \|a_k - a_k'\|^2 + \beta \|b_k - b_k'\|^2 \tag{3.23a}$$

$$\text{s.t. } \forall j,\ a_k^T p_k^j \leq b_k, \tag{3.23b}$$

$$\forall j,\ a_k^T p_{k+1}^j \leq b_k, \tag{3.23c}$$

$$\forall i,\ a_k^T r_k^i \geq b_k + d, \tag{3.23d}$$

$$\forall i,\ a_k^T r_{k+1}^i \geq b_k + d, \tag{3.23e}$$

$$-\begin{bmatrix}1\\1\\1\end{bmatrix} \leq a_k \leq \begin{bmatrix}1\\1\\1\end{bmatrix}, \tag{3.23f}$$

$$1 - \varepsilon \leq a_k^T a_k^p \leq 1 \tag{3.23g}$$

Then, we use these separating planes to compute the collision-free robot trajectory with a QP:

$$\min_{\mathbf{u}} \; \sum_{k=0}^{N-1} \|x_{k+1} - x_{k+1}^{des}\|_Q^2 + \|u_k - u_k^{des}\|_R^2 \tag{3.24a}$$

$$\text{s.t. } \forall k \in \{0,\dots N-1\},\ \underline{u} \leq u_k \leq \overline{u}, \tag{3.24b}$$

$$\forall k \in \{1,\dots N\},\ \underline{q} \leq q_k \leq \overline{q}, \tag{3.24c}$$

$$\forall k \in \{1,\dots N-1\},\ \underline{\dot{q}} \leq \dot{q}_k \leq \overline{\dot{q}}, \tag{3.24d}$$

$$\dot{q}_N = 0, \tag{3.24e}$$

$$\forall k \in \{0,\dots N-1\},\ \forall i,$$
$$a_k^T r^i(q_k^p) + a_k^T J(q_k^p)(q_k - q_k^p) \geq b_k + d_{safe}, \tag{3.24f}$$

$$\forall k \in \{0,\dots N-1\},\ \forall i,$$
$$a_k^T r^i(q_{k+1}^p) + a_k^T J(q_{k+1}^p)(q_{k+1} - q_{k+1}^p) \geq b_k + d_{safe} \tag{3.24g}$$

where the sequence of states $(q_k, \dot{q}_k)$ is linearly related to the control sequence $\mathbf{u}$ through (3.20).

Algorithm 1: CollisionFreeTrajectoryComputation

Input: $\mathbf{u}^p$, $\mathbf{q}^p$, $\mathbf{a}^p$, $\mathbf{b}^p$
Output: $\mathbf{u}$
i = 0 ;
do
  $\mathbf{u}^p = \mathbf{u}$;
  Update Robot's parameters ;
  {**a**,**b**} ⟵ Solve QPs (3.23) ;
  {**u**} ⟵ Solve QP (3.24) ;
  i ++;
while *(*$||\boldsymbol{u} - \boldsymbol{u}^p||^2$ *OR* $i \leq MAX\_STEP$*)*;

The separating problem optimisation problem and collision-free robot trajectory optimisation

problem is indeed a linear approximation of the nonlinear problem :

$$\min_{\mathbf{u},\mathbf{a},\mathbf{b},\mathbf{d}} \sum_{k=0}^{N-1} \|x_{k+1} - x_{k+1}^{des}\|_Q^2 + \|u_k - u_k^{des}\|_R^2 + \tag{3.25a}$$

$$\sum_{k=0}^{N-1} -d_k + \alpha d_k^2 + \beta \|a_k - a_k'\|^2 + \beta \|b_k - b_k'\|^2 \tag{3.25b}$$

$$\text{s.t. } \forall k \in \{0,\dots N-1\},\ \underline{u} \leq u_k \leq \overline{u}, \tag{3.25c}$$

$$\forall k \in \{1,\dots N\},\ \underline{q} \leq q_k \leq \overline{q}, \tag{3.25d}$$

$$\forall k \in \{1,\dots N-1\},\ \underline{\dot{q}} \leq \dot{q}_k \leq \overline{\dot{q}}, \tag{3.25e}$$

$$\dot{q}_N = 0, \tag{3.25f}$$

$$\forall k \in \{0,\dots N-1\},\ \forall i,$$
$$a_k^T r^i(q_k) \geq b_k + d_{safe}, \tag{3.25g}$$

$$\forall k \in \{0,\dots N-1\},\ \forall i,$$
$$a_k^T r^i(q_{k+1}) \geq b_k + d_{safe}, \tag{3.25h}$$

$$\forall k \in \{0,\dots N-1\},\ \forall j,\ a_k^T p_k^j \leq b_k, \tag{3.25i}$$

$$\forall k \in \{0,\dots N-1\},\ \forall j,\ a_k^T p_{k+1}^j \leq b_k, \tag{3.25j}$$

$$\forall k \in \{0,\dots N-1\},\ \forall i,\ a_k^T r_k(q_k)^i \geq b_k + d, \tag{3.25k}$$

$$\forall k \in \{0,\dots N-1\},\ \forall i,\ a_k^T r_{k+1}(q_{k+1})^i \geq b_k + d, \tag{3.25l}$$

$$\forall k \in \{0,\dots N-1\},\ \|a_k\| = 1, \tag{3.25m}$$

Solving the above nonlinear problem with a nonlinear solver is fairly slow. We could use an alternative method to solve (3.25). We first compute a given set of plane locations by solving (3.23) with the variable $\mathbf{u}$ fixed, then, use the set of planes to solve the problem (3.24) with planes fixed. Solving problem (3.24) gives new values of the planes positions to compute the next iteration and repeat this process until convergence, as shown in algorithm 1.

3.5 Simulations results

The proposed MPC framework in Fig. 3.3 has been implemented on a Franka Emika Panda robot and simulated with ROS, shown in Fig. 3.5. This 7-DOF manipulator robot was specifically designed for human-robot interactions [Emika]. The Franka Emika has a payload of 3 kg for a weight of 17.8 kg. Each axis has force/torque sensors that provide information about the forces applied to the robot. Using this information, the robot can interact with its environment and can detect potential collisions [Haddadin2008]. The robot provides low-level programming based on C++ and Python APIs, or high-level programming based on web GUI, and is now widely used in academic research and industry. In the next section, we will look at different scenarios, such as the robot avoiding static and dynamic obstacles while performing pick-and-place tasks.

3.5.1 Tasks

The robot is performing a pick-and-place task between two positions $G_A = \begin{pmatrix} 0.4 & -0.4 & 0.25 \end{pmatrix}^T$ and $G_B = \begin{pmatrix} 0.4 & 0.4 & 0.25 \end{pmatrix}^T$ as shown in Fig. 3.6. Recall our objective function of MPC scheme

Figure 3.5: Visualisation of Franka Emika Panda robot in RVIZ.

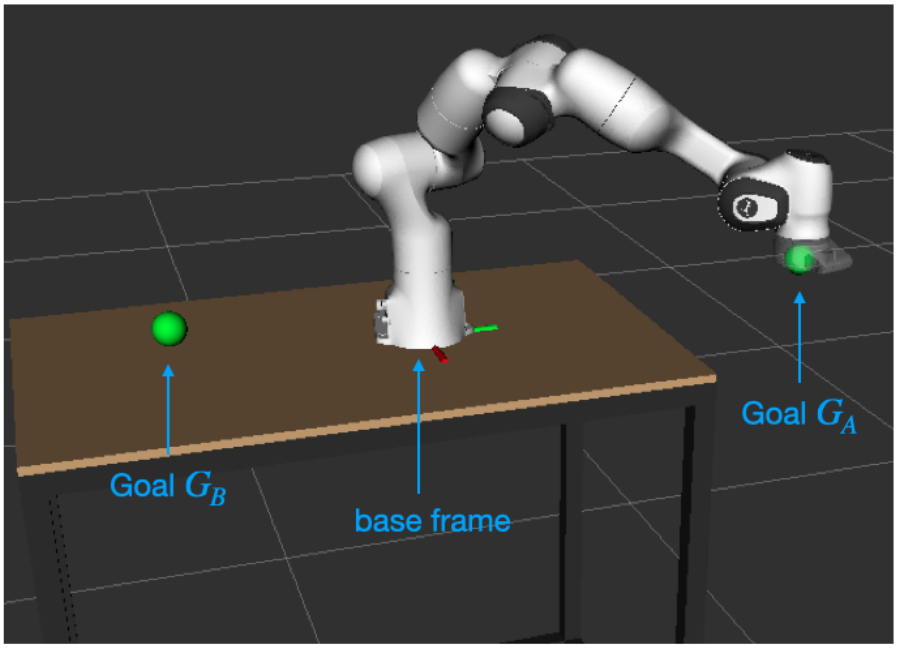


Figure 3.6: Two green spheres represent the desired pick-and-place positions for the manipulator.

is defined in Eq. (3.5.1) :

$$\underset{\mathbf{u}}{\text{minimize}} \sum_{k=0}^{N-1} \|x_{k+1} - x_{k+1}^{des}\|_Q^2 + \|u_k - u_k^{des}\|_R^2$$

The first term minimizes the error between the current state of the robot and the desired state, for a point-to-point task we can have zero desired velocity, which gives $x^{des} = \begin{bmatrix} G^T & 0 & 0 & 0 \end{bmatrix}^T$. The second term is used for regularization when we keep the desired u to zero. For the weighted parameters Q and R, we will give the ratio as $Q/R = 10e^6$. Then, we will rewrite these term into quadratic form to be solved with any off-the-shelf solvers such as [Ferreau2014]:

$$
\begin{aligned}
&\sum_{k=0}^{N-1} \|x_{k+1} - x_{k+1}^{des}\|^2 + \|u_k\|^2 \\
&= \sum_{k=0}^{N-1} \| \underbrace{P_u}_{E_k} u_k + \underbrace{P_x x_k - x_{k+1}^{des}}_{f_k} \|^2 + \|u_k\|^2 \\
&= \sum_{k=0}^{N-1} \frac{1}{2} u_k^T \underbrace{E_k^T E_k}_{H_k} u_k + u_k^T \underbrace{E_k^T f_k}_{g_k} + u_k^T I_n u_k \\
&= \frac{1}{2} \mathbf{u}^T H \mathbf{u} + \mathbf{u}^T g
\end{aligned}
$$

with,

$$
\mathrm{H} \in \mathbb{R}^{Nn \times Nn} = \begin{bmatrix} E_0^T E_0 + I_n & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & E_{N-1}^T E_{N-1} + I_n \end{bmatrix} \qquad g \in \mathbb{R}^{Nn} = \begin{bmatrix} E_0^T f_0 \\ \vdots \\ E_{N-1}^T f_{N-1} \end{bmatrix}
$$

3.5.2 Constraints

Panda's physical constraints								
Name	Joint 1	Joint 2	Joint 3	Joint 4	Joint 5	Joint 6	Joint 7	Unit
q_{max}	2.897	1.763	2.897	-0.070	2.897	3.752	2.897	rad
q_{min}	-2.897	-1.763	-2.897	-3.072	-2.897	-0.018	-2.897	rad
$\dot{q}_{max}$	2.175	2.175	2.175	2.175	2.610	2.610	2.610	rad.s^{-1}
$\ddot{q}_{max}$	15	7.5	10	12.5	15	20	20	rad.s^{-2}

The physical constraints associated with Franka Emika Panda are summarized in the Tab. (3.5.2). As in the previous section, the position, velocity and acceleration constraints are stacked in a vector and matrix to group the constraints over the prediction horizon. We get inequality constraints in the form:

$$\mathbf{lb} \leq \mathbf{u} \leq \mathbf{ub} \tag{3.26}$$

$$\mathbf{lbA} \leq \mathbf{Cu} \leq \mathbf{ubA} \tag{3.27}$$

3.5.3 Task achievement in static environment without obstacle

In Figure (3.6), two green spheres represent alternate desired positions for the manipulator's end-effector. When the environment is static and without obstacles, the only constraints to be considered are the physical constraints of the robot, as shown in Tab. (3.5.2). The choice of the prediction horizon is based on the deceleration capacity of the different joints. For example, when the first joint is at maximum speed with maximum acceleration, the robot needs about 0.15 s to stop and for the second joint the stopping time is about 0.3 s. To satisfy the terminal constraint Eq. (3.21e) which imposes that the robot is at rest at the end, the prediction horizon must therefore be longer than 0.3 s. In this experiment, we choose for a prediction horizon of length 0.5 s with Δt=0.1 s, then N = 5.

Figure 3.7 gives a visualization of the generated trajectory for the next N time steps. Each via-point is represented as a Cartesian frame with the x axis in red, the y axis in green and the z

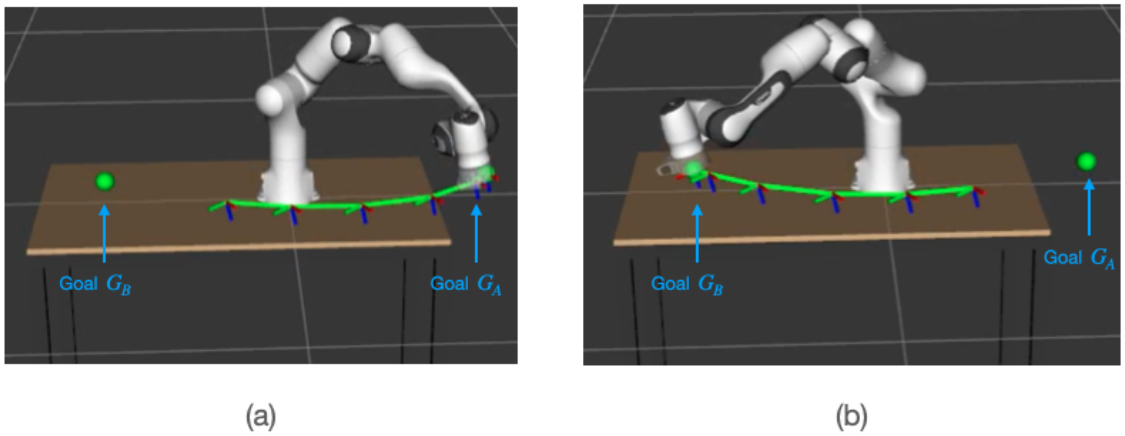


Figure 3.7: The robot moves towards the goal, the frames represent the next N points generated on the prediction horizon.

axis in blue. Figure 3.8 shows the first four joint configurations and velocities, since the last three are only useful for orientation. We can see that the first joint is working at full speed (2.1 rad.s^{-1}) and the Cartesian velocity on the y-axis reaches to 2 $m.s^{-1}$ allowing the robot to move extremely fast.

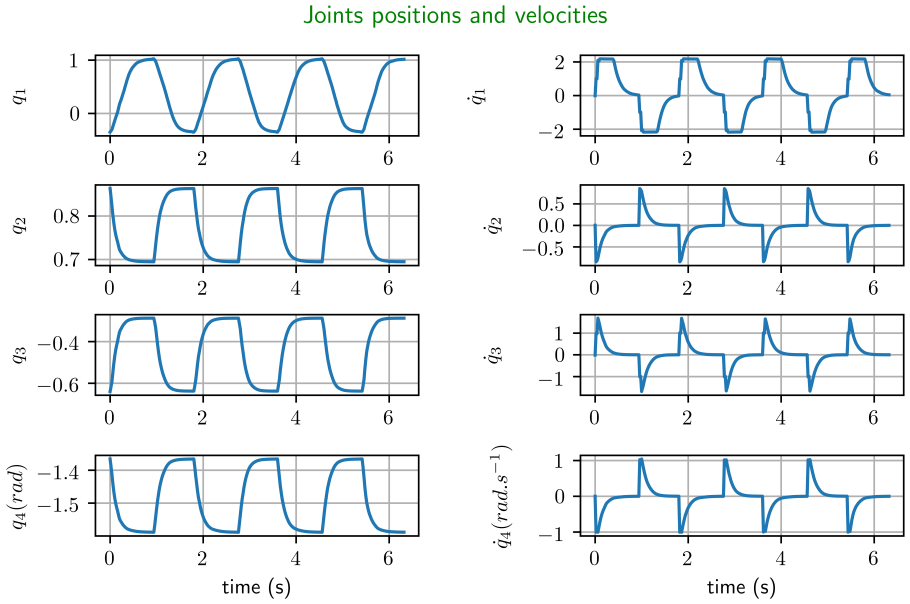


Figure 3.8: Joint positions and velocities.

3.5.4 Task achievement in a dynamic environment

In this section, we test our trajectory generator in a dynamic environment with an obstacle that makes a circular motion and interferes with the robot's task. In Figure 3.10, we get the view of the environment in the XY plane. The obstacle is represented by a cylinder with a diameter of 10 cm and a length of 50 cm, and the yellow circle represents its trajectory.

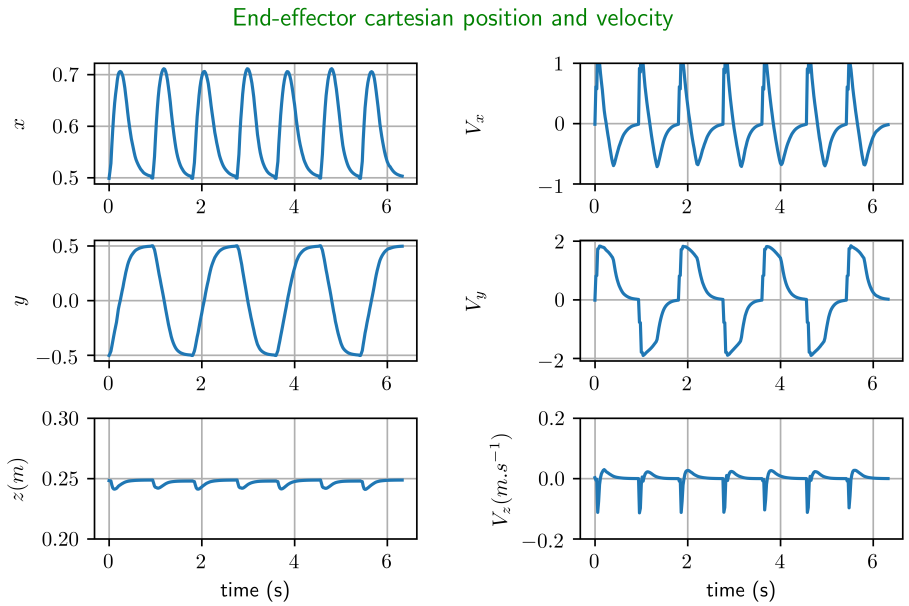


Figure 3.9: End-effector Cartesian positions and velocities.

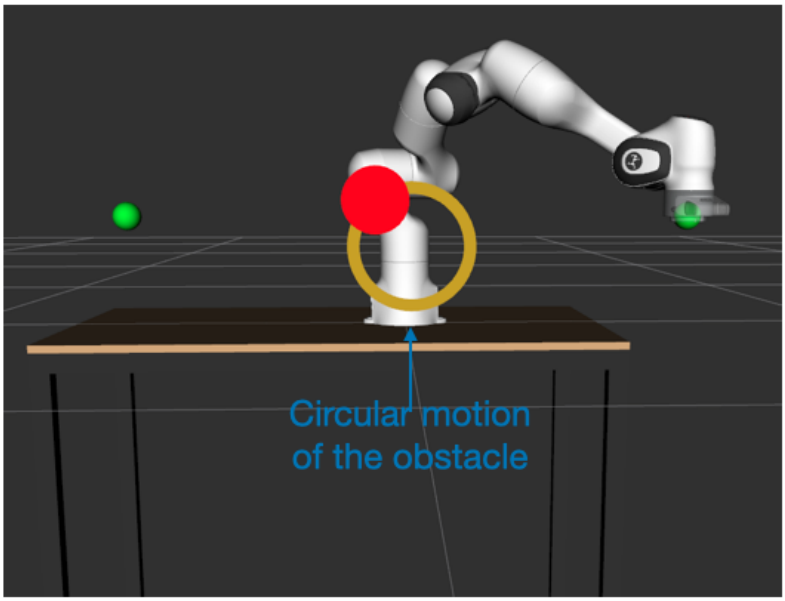


Figure 3.10: Circular motion of a cylinder shape obstacle.

Choice of the horizon of the prediction

The terminal constraint (3.21e) imposes that the robot is at rest at the end of the prediction horizon in order to provide a passive motion safety guarantee. However, the prediction horizon must be chosen appropriately, otherwise, two major problems can occur. They are 1) If the prediction horizon is too short, the robot loses its performance due to the terminal constraint. 2) If the prediction horizon is too long, the estimation of the person's motion may not be accurate at all.

- Lower limit of the prediction horizon: The terminal constraint forces the robot to have zero velocity at the end of the prediction horizon. Therefore, the robot's velocity will depend on the maximum deceleration capacity. A robot operating at maximum velocity needs $T_{low} =$

$\frac{V_{max}}{a_{max}}$ seconds to decelerate to zero velocity. Consequently, to enable the robot to move fast, the prediction of the horizon must be greater than T_{low}. For example, the first joint of the robot at full speed needs about 0.15 s to stop and for the second joint the stopping time is 0.3 s. So T_{low} is longer than 0.3 s.

- Upper bound of the prediction horizon: On the other hand, optimal collision-free trajectory depends on how long the human motion can be predicted. It was shown in [Kimm2015b] that the human's hand motion can be very fast, $8.1 \pm 1.4\ m.s^{-1}$ for males and $6.6 \pm 1.6\ m.s^{-1}$ for females in extreme cases. A robot is not able to avoid the collision in this situation. However, in a normal grasping situation, the human hand moves with maximum velocity of about 1 $m.s^{-1}$ and an acceleration of 5 $m.s^{-2}$ [Grujic2015b]. An appropriate upper bound can be chosen as 0.3 to 0.5 s since human motion prediction quickly becomes imprecise when the prediction horizon is longer than 0.5 s [Toyer2017, Kratzer2019].

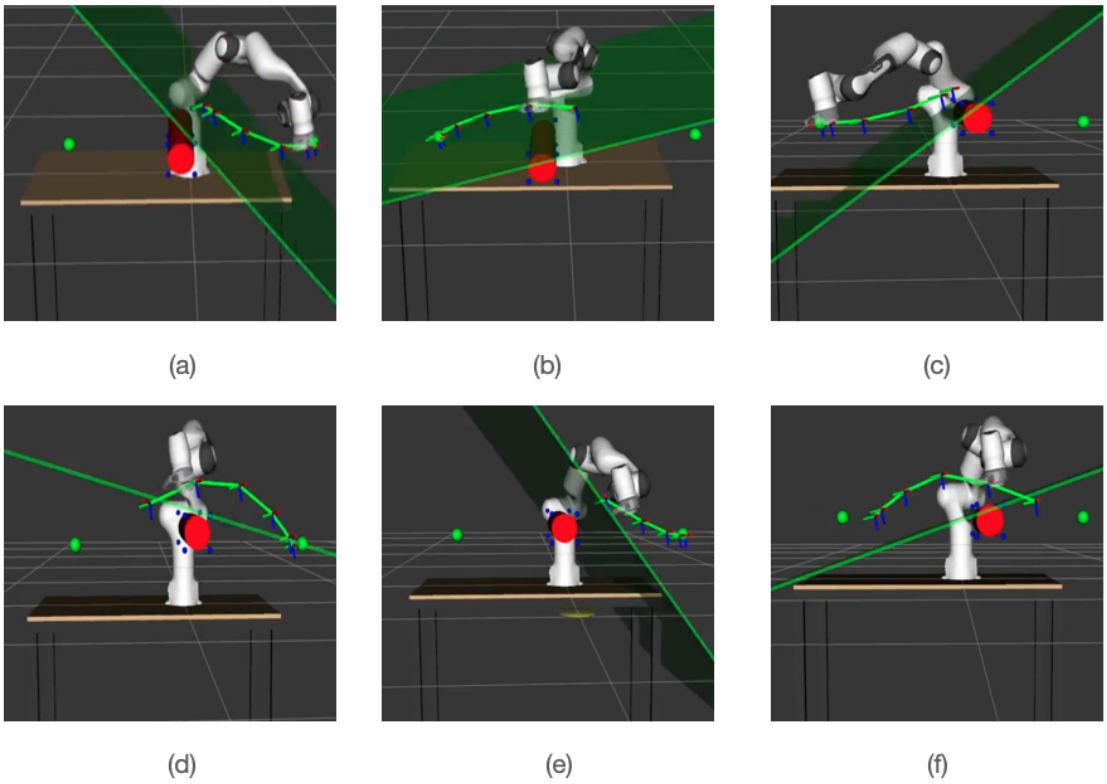


Figure 3.11: Snapshots of simulation. (a)-(c) The robot deviates its initial trajectory to avoid potential collision with obstacle. (d)-(f) The robot change its trajectory according to the motion of the obstacle to enable safe task accomplishment.

Figure 3.11 shows the snapshots of the simulation. In this scenario, we choose a prediction horizon of length 0.5 s with Δt = 0.1 s, N = 5 and the safety distance is set to 20 cm. The resulting QPs (3.23) and (3.24) are solved with qpOASES [Ferreau2014]. The robot performs the same task as shown in Figure 3.7 while avoiding collision with a moving obstacle. The obstacle is represented as a cylinder with 0.1 m for diameter and 0.5 m for length, it can be considered as a human arm. The obstacle follows a circular trajectory as shown in Figure 3.10 with a rotation velocity of 1 $rad.s^{-1}$. In Figure 3.11(a)-(f), the robot deviates from its initial trajectory as described in section 3.5.3. This trajectory deviation is also justified by a significant variation of the end-effector cartesian velocity in the z-axis as shown in Figure 3.13. Table 3.2 summarises the size and the

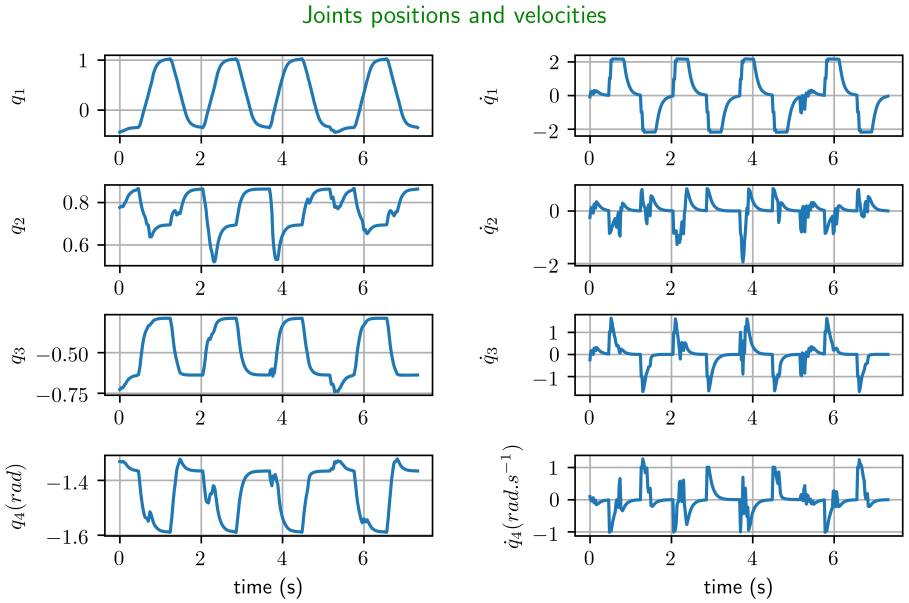


Figure 3.12: Joint positions and velocities in the presence of a dynamic obstacle.

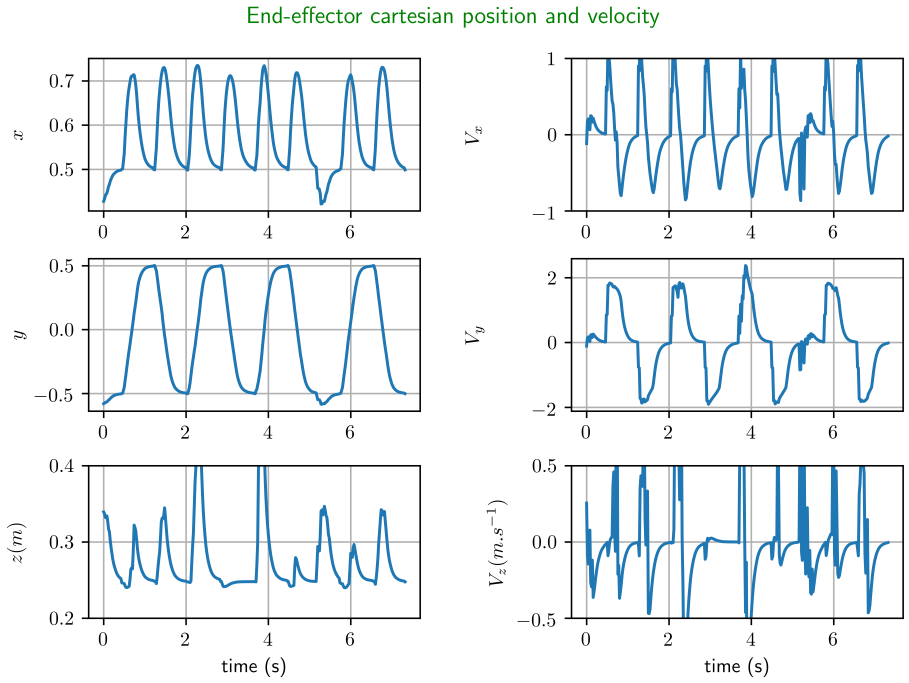


Figure 3.13: End-effector Cartesian positions and velocities in the presence of a dynamic obstacle.

computing time of separating plane problem and trajectory generation problem. It shows that the two quadratic programming problems can be solved efficiently for real-time applications such as human-robot collaboration.

It can happen that the goal is blocked by the obstacle as shown in Figure 3.14. The robot will stay as close to the goal as possible while maintaining the imposed safety distance and the constraint. Once the obstacle is out of the way, the robot continues its trajectory to complete the task.

	Separating plane (Eq. 3.23)	trajectory generation (Eq. 3.24)
Number of variables	5	35
Number of constraints	21	86
Average solving time (s)	0.00067	0.00612

Tableau 3.2: Problem size and computation time for separating plane and trajectory generation problem

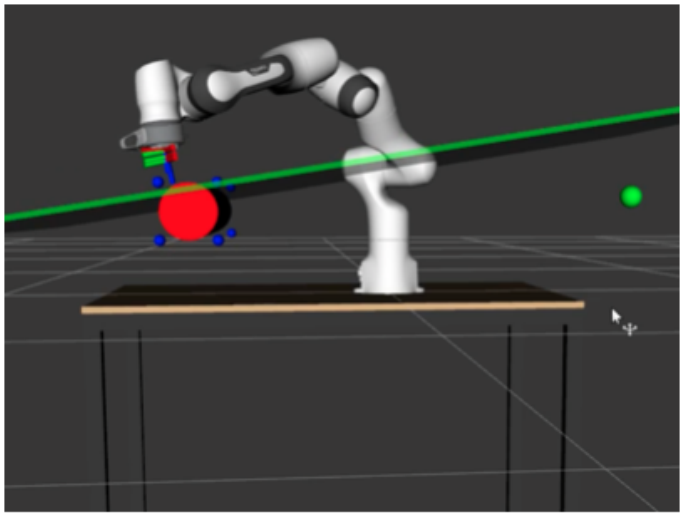


Figure 3.14: The obstacle blocked the goal

3.6 Conclusion

This chapter presents the general Model Predictive Control framework for safe trajectory generation, as outlined in Figure 3.3. We give mathematical details on how to formulate the robot's task and constraints in the MPC framework in Section 3.2 and Section 3.3. Simulation results on trajectory generation in the free workspace and in the workspace with dynamic obstacles are described in Section 3.5. However, the human model presented in the simulation is simplified into a cylinder geometry and the prediction of the motion is known in advance. In the next chapter 4, we will complete the perception module by adding human body detection and human hand motion prediction.

4

Human detection and motion prediction

Contents

In the previous chapter, we introduced our MPC framework for generating a collision-free trajectory in the presence of dynamic obstacles. In fact, collision avoidance is equivalent to finding a plane that separates the robot and the object, which are located in the same space. Thus, the knowledge of the object's position has a strong impact on the performance of the trajectory generation. Two methods for human and posture detection have been widely considered in the literature: vision based methods using a camera and inertial measurement units (IMUs) based on a special suit for motion capture as shown in Fig 4.1. The latter is not suitable for collaborative applications due to the need to wear a special uniform with sensing devices and is relatively complex, expensive, and difficult to maintain.

This is dedicated to describe the perception module of the MPC framework. The goal is to detect the position (x,y,z) of a person potentially entering the robot's workspace using a single RGB-D camera and to predict his or motion over a short time horizon. One challenge is that human operators tend to follow different motion patterns, depending on several factors such as intentions and the structure of the environment. Locating a person in an image and extracting their key points is a challenging task due to occlusion, blur, illumination, rotation and scale variations. Another challenge is that human operators tend to follow different motion patterns, depending on several factors such as intentions and the structure of the environment.

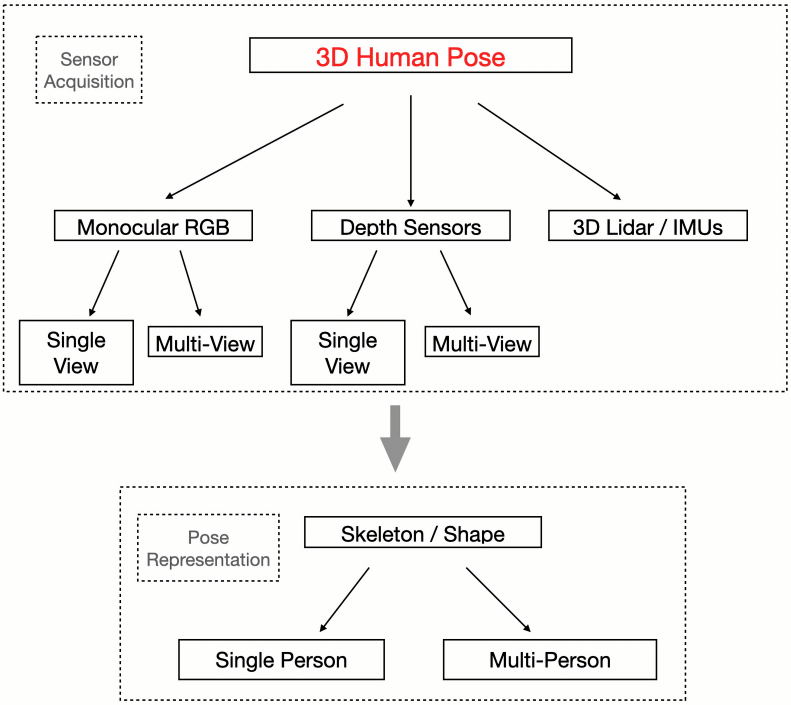


Figure 4.1: A simplified taxonomy of the different sensors used for 3D human pose estimation and pose representation

In the following, we will first discuss the state of the art on human pose detection and motion prediction in Section 4.1. Pose detection based on an RGB-D camera is described in Section 4.2. The hand motion prediction is explained in Section 4.3 and the results are presented in Section 4.4.

4.1 State of art

Before we start discussing how to obtain the position of a person in a 3D environment, let us briefly compare the different sensors that allow us to obtain a representation of the environment. As shown in Figure 4.1, we can obtain a representation of the environment using a monocular camera, a depth camera, or with 3D lidar. In addition, IMU systems for motion capture provide a direct representation of the person's pose. As we discussed before, the inertial sensor based method is not suitable for collaborative tasks and the 3D lidar is relatively expensive. The vision sensors method offers the choice of using a single camera or multiple cameras. The advantage of multi-view is that it can provide more accurate position and robust results to the occlusion, but the processing and fusion of data remains relatively complex. In this thesis, we focus on the use of the RGB-D camera which is a combination of single view from monocular and depth sensors. This provides a balance between simplicity and performance, for a more detailed comparison, we suggest consulting the review written by [Wang2021].

Once the right sensor has been selected for data acquisition, the next step is to obtain the information of the detected human between the skeleton and the shape representation as shown in Figure 4.1. The skeleton representation gives the key-points location of different joints as shown in Figure 4.2(a) and the shape representation gives the contour of the human body as shown in the bottom left image in Figure 4.2(b). For the collaborative task, the contour information is not essential, the location of the key points is sufficient to inform us about the pose of the people and then give the robot the necessary information to avoid collisions. In the following, we will describe several different methods for 3D human pose estimation based on RGB-D camera. The

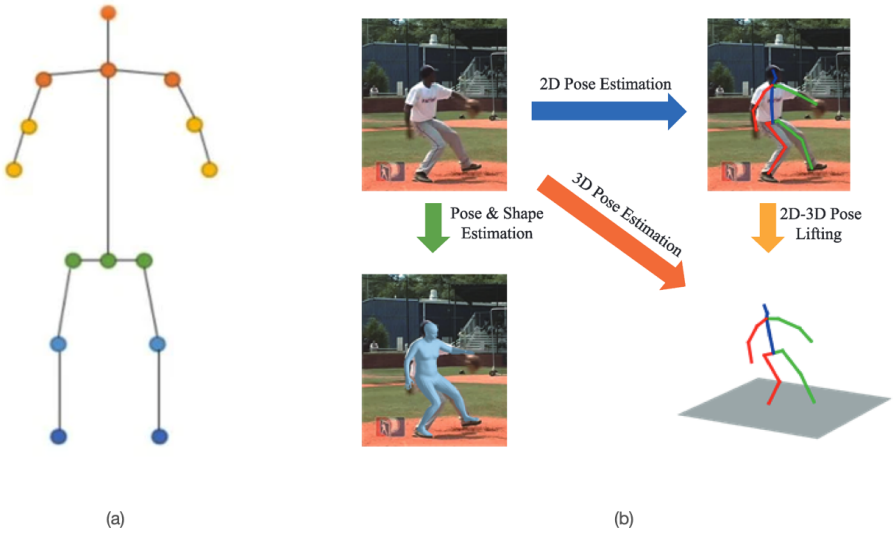


Figure 4.2: Illustration of human pose representation and pose estimation. a) Representation of the human body joints with 15 key-points; b) 3D human pose estimation, 2D-3D pose lifting and human pose and shape estimation.

depth camera based on time-of-flight (TOF) and structured light technologies [Sarbolandi2015] can provide the distance information from a single depth image or a point cloud, thus, thus gives the possibility to deal directly with 3D data. However, human pose estimation has some unique characteristics and challenges such as flexible body configuration indicates complex interdependent joints, diverse body appearance and complex environment may cause occlusion. The existing works can be divided into three categories such as template-based method, feature-based method and learning-based method according to [Xu2021].

The template-based method compares the similarity between the detected object and the con-

structed template to identify the motion category. [Zhu2010] proposes a head-neck-trunk deformable template represented by circles, trapezoids and rectangles. Mathematical template such as Gaussian Mixture Mode (GMM) is applied by [Ye2014] to simultaneously extract pose and shape of an articulated object from a single depth camera. However, template-based method needs to establish a template library of parameterized template to compare with human body which is time-consuming, and the accuracy of template-based methods is very limited due to the diversity of the different human pose in space. Feature-based methods uses geodesic distance information [Yuan2017], geometric feature such as silhouette [Xu2020] to estimate human joints. Feature-based template has some disadvantages, for example, it requires prior knowledge to combine with extracted global or local features to obtain the 3D pose and it is not suitable for changing poses which is necessary to further optimize the robustness.

The learning-based method uses the network structure to automatically learn the required features from the input data and then the learned features can be further used to extract the human pose. Compared to the above two methods, the results can achieve higher accuracy by learning in a large dataset and it is robust to scale processing. [Shotton2012] developed two algorithms trained on a large dataset with decision forests and depth-invariant image features to effectively detect different human poses. Recently, deep learning structure has become a very active research area [Zhou2020], [Haque2016] trains neural networks based on convolutional and recurrent architecture to achieve depth image human pose estimation. In [Wang2020], the author directly uses a sequence of point clouds as the input of a neural network to output the 3D human pose, because point clouds can provide more geometry information than depth images. However, training such a model requires a special dataset, which is not easy to construct, and the running time may not be well suited for real-time application. Therefore, as a more accessible approach, estimating human pose from RGB images captured by regular cameras and then transforming the 2D information into 3D pose is attracting the attention of researchers.

The simplest method for estimating 3D human poses is to design an end-to-end network to predict the 3D key-point locations directly from 2D images, as shown in Figure 4.2(b), where the input data is a color image and the output of the model is the human joint locations or 3D mesh, without any intermediate representation. [Li2014] directly estimates the joint locations relative to the root joint locations, and [Kanazawa2018] infer 3D mesh parameters directly from image features in an end-to-end manner. [Zhou2016] embeds a kinematic object model as prior knowledge into the deep network model for general articulated object pose estimation.

The second method is the process of 2D-3D lifting as shown in the right side of Figure 4.2(b), this consists of two-stage based cascaded frameworks that first performs 2D pose estimation to predict 3D joint positions in the a color image with 2D pose estimator and the lifts these 2D joints to the 3D space. The key idea is that 2D pose estimation can be easily performed due to the availability of large-scale datasets with 2D annotations in the wild. Examples include two of the state-of-the-art 2D pose estimators such as Mediapipe [Zhang2020a] and Openpose [Cao2017]. MediaPipe is a an open-source system from Google and OpenPose is a multi-person human key point detection system, both can achieve single image pose detection for real-time application. Benefiting from the reliable performance of state-of-the-art 2D pose detectors, the 2D-3D pose lifting process generally outperforms the methods that directly regress 3D poses from images. [Martinez2017b] focused on lifting 2D poses to 3D by applying two fully connected layers with a residual connection to 2D to 3D key-point regression. [Moreno-Noguer2017] proposed to represent 2D and 3D poses with N $\times$ N matrices of Euclidean distances between each pair of joints and formulated the 3D pose estimation problems as a 2D-to-3D distance matrix regression. However, recovering a 3D human pose from a single image suffers from the ill-defined problem that different 3D poses can correspond to the same 2D images. While it is easy to obtain manual 2D pose annotations for training datasets, it is difficult to collect accurate 3D pose annotations is difficult, and the lack of real-world benchmark datasets makes the problem of 3D pose estimation very challenging. However, since we have the depth information from the RGB-D camera, we can effectively combine

the 2D pose locations with depth images to obtain 3D pose locations. This idea is also explored in the recent work of [Docekal2022] for the close-up human-robot interaction. They convert the 2D positions of human key points into 3D by adding the depth information. This method is also called naive-lifting in the study of [Zimmermann2018], which gives a pretty good result compared to it's simplicity.

In this thesis, we formulate our perception module for human pose detection in two parts: 1) Detection of the human pose in Cartesian space as described above. 2) Prediction of the human's motion in a horizon prediction, which will be presented in the next section.

Human motion prediction is a more complex task. Existing prediction methods can generally be divided into model-based and learning-based methods. Model-based methods attempt to directly model the human kinematics or dynamics to find the corresponding arm motor control. According to the human motor control literature, human movements follow an optimal feedback control strategy that links motor behavior, limb mechanics and neural control as described in [Scott2004]. However, choosing the optimal trajectory cost is not trivial because the human musculoskeletal system has more DOFs than are necessary to perform a given task. This kinematic, dynamic, and actuation redundancy problem is not straightforward in terms of the equations of motion. Numerous cost functions have been identified in the literature. In [Flash1985], the authors model the point-to-point kinematic motion of the hand with a minimum Cartesian jerk (third derivative of the Cartesian coordinates) for an arm motion in the horizontal plane. The authors in [Uno1989] include dynamics with a minimum torque change model in the horizontal plane, but the results have not been validated for 3D motion. While it is difficult to manually define these motor control criteria, the author in [Sylla2014a] defines a combination of seven different criteria (such as Cartesian jerk, angular jerk, angular acceleration, torque change, torque, geodesic and energy) and an inverse optimization method was used to find the weight associated with each criterion. In [Pereira2017b], instead of artificially finding arm motor control, the authors overapproximate arm occupancy with a maximum velocity model, but this can be too restrictive when the prediction horizon is long. There are problems with these approaches. First, human dynamics are highly nonlinear and non-deterministic; they can vary with emotions and physical state, so direct modeling can be quite inaccurate in different situations. Second, the hypothesis of the human rationality is often invalid, so the construction of optimization criteria based on this hypothesis can be very ambiguous and the combination of the different criteria seems to be chosen manually.

Human motion is the result of complex biomechanical processes that are difficult to model. As a result, state-of-the-art works in motion prediction focuses on data-driven models [Bishop2006]. A Gaussian Mixture Model (GMM) has been used in [Mainprice2013b] to regress collected trajectories into different clusters in the offline stage. The encoded human motion library is then used in the online stage to predict the future human motion. A probabilistic framework such as Hidden Markov Models (HMM) [Ding2011c] can be used to predict a safety-critical region that the worker may occupy during the human-robot collaboration task. However, the continuous distribution of trajectories cannot be fully represented with HMM by transition probabilities, and they are difficult to generalize to new environments.

Recent work on short-term human motion prediction has focused on recurrent neural network (RNNs) because of their ability to handle sequential data, as described in Section 2.5.3. Because of their internal memory structure, RNN's can remember important things about the input they have received, which allows them to be accurate in predicting the output. In [Wang2017], the authors present a framework from vision-based hand movement prediction in a human-robot collaboration scenario to generate a collision-free trajectory. In their approach, a pre-trained CNN model is used to extract visual features from the video input sequence, and a Long-Short Term Memory (LSTM) model (a variation model of RNN) is trained to predict the hand movement. However, since the input is the image, they only predict the relative displacement of the hand in 2D space. In [Zhang2020b], an RNN-based model incorporating uncertainties has been used to predict human trajectory in collaborative assembly of an automobile engine.

This goal of this chapter is to develop our perception module presented in Figure 3.3 of Chapter 3. In order to generate a collision-free trajectory, our MPC controller must know the location and future positions of an operator. Without loss of generality, and to illustrate the idea, we will only predict the future positions of the hand, since it is the part of the body that enters the robot's workspace the most and can potentially have direct contact with the object being handled by the robot. The structure of the rest of the chapter is to present these different components:

1. Define human pose detection in a RGB image in Section 4.2, which gives the human pose location in 2D pixel levels.

2. Extract the corresponding hand's Cartesian 3D position of the corresponding hand by combining the knowledge of the depth image with the 2D pixel position of the hand using the pinhole camera model as described in Section 4.2.2.

3. In Section 4.3, we define an LSTM neural network architecture to predict the hand's future step positions, which will allow our MPC controller to generate a collision-free trajectory.

4.2 Human 3D pose detection

In Section 4.1, several methods for human pose estimation in 3D space with different sensors have been presented, and some of which are computationally expensive and not suitable for real-time applications. In fact, we use an RGB-D camera which provides a good balance between simplicity and performance, to localize human 3D poses in a human-robot collaboration environment. In the following section, we describe the detection workflow in detail: 1) extract human keypoints from an RGB image. 2) mapping to depth camera to obtain 3D point locations. Due to the limitation of the monocular camera, the problem of occlusion is inevitable, so not every keypoint is detectable. Therefore, only the keypoints of the person's left arm are considered.

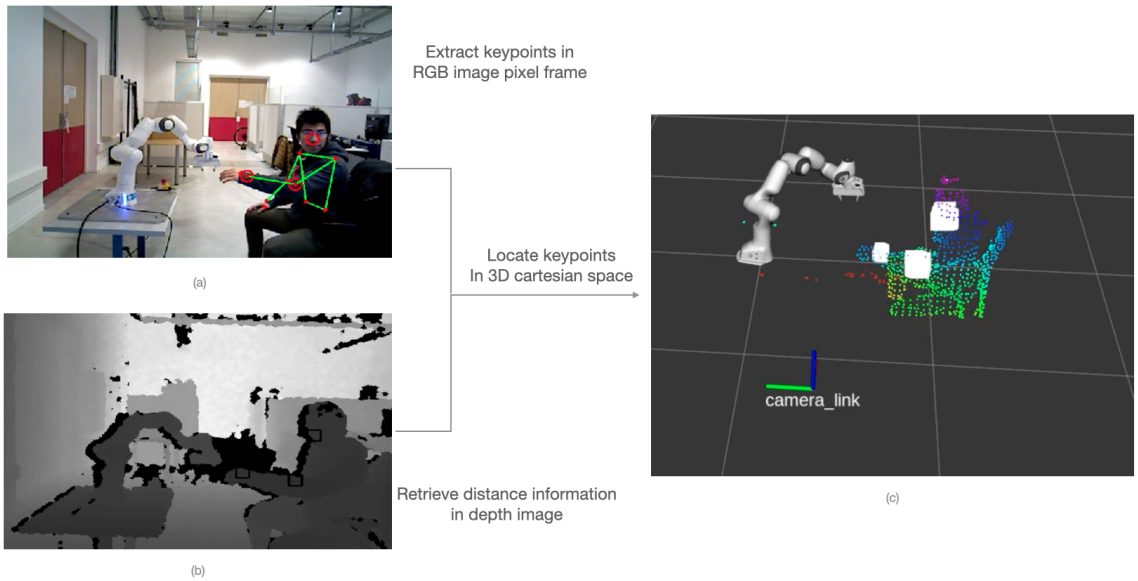


Figure 4.3: Human keypoints detection from RGB-D camera by combining colour and depth image

4.2.1 Human keypoints in RGB images

The convolution neural network (CNN) presented in Section 2.5 is very efficient for handling structured data such as images. Through the CNN architecture, the first layer of convolutions learns some basic features (such as edges and lines), the next layers learn features that are more complex (circles, squares, etc...), the following layers find even more complex features (such as parts of the face, a hand, a trunk, etc ...), and so on. In [Zeiler2014], the authors give a great visualization of features learned from different layers of a CNN architecture, but mostly limited to the first layer where projections to pixel space are possible.

Building a human pose estimation algorithm from scratch is not only time consuming but also very expensive. First, it must select and develop appropriate learning algorithms and models, second, collect a massive dataset to enable the models to learn features through. Finally, the resource consumption must be balanced against the quality of the solutions. However, there are several pre-trained models available in opensource, such as Mediapipe and Openpose presented in Section 4.1. Furthermore, these models have been well tested on different datasets and they are well optimized to be deployed in real-time applications.

In this thesis we use Mediapipe as our human pose estimation model because Mediapipe provides cross-platform solutions and the Python version of mediapipe can be easily modified to integrate it into Robot Operating System (ROS). Figure 4.3(a) shows the keypoints location of the upper-body in an RGB image is depicted. We will give more implementation details about Mediapie in ROS in Chapter 5.

4.2.2 Human keypoints in 3D space

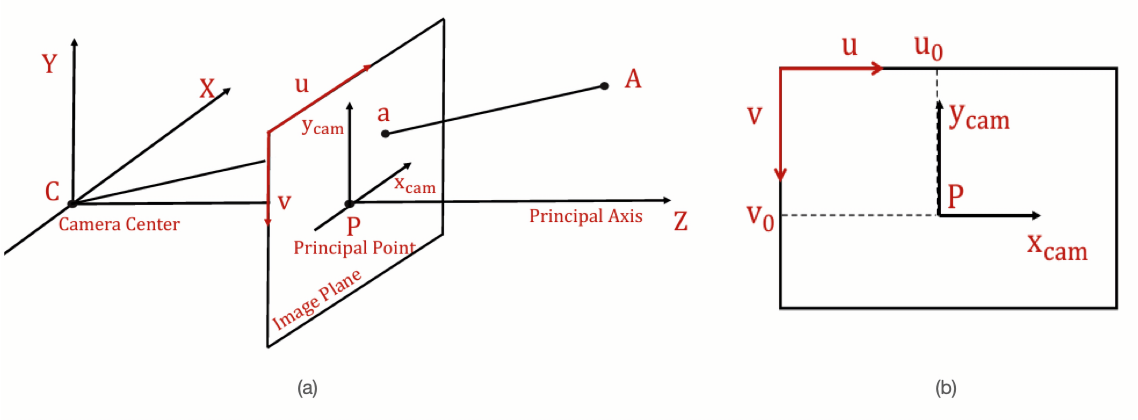


Figure 4.4: Pinhole camera model:(a) Projection of camera frame to image plane frame. (b) THe origin of image plane moved to (v,u) with offset v_0 and u_0.

With an RGB image as Mediapipe's input, we get as the output the position of the keypoints in the image frame (i.e. expressed in pixels). To perform 2D to 3D conversion, we first need to transform our keypoints expressed in image frame to camera center frame, and then add depth information by pinhole camera model. The pinhole camera model describes the mathematical relationship between the coordinates of a point in 3D space and its projection onto the image plane, as shown in Figure 4.4(a). The mathematical expression for this projection is:

$$\begin{pmatrix} X, & Y, & Z \end{pmatrix}^T \Rightarrow \begin{pmatrix} u, & v \end{pmatrix}^T = \begin{pmatrix} \frac{f_x X}{Z} + u_0, & \frac{f_y Y}{Z} + v_0 \end{pmatrix}^T \tag{4.1}$$

Where $(X,Y,Z)^T$ are the Euclidean coordinates of a point. $(u,v)^T$ is the project coordinate of the 3D point in the image frame. f is the focal length, the distance between C (camera center) and

P (principal point). $(u_0, v_0)^T$ is the offset from the principal point to the upper left corner of the image plane. In matrix form, we obtain:

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & u_0 & 0 \\ 0 & f_y & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{4.2}$$

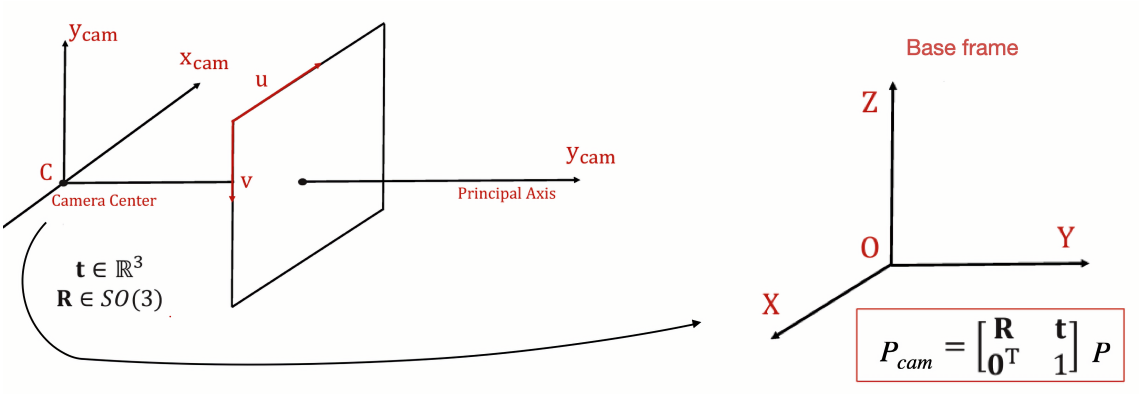


Figure 4.5: Homogenous transformation between the camera frame and base frame.

Note that in Figure 4.4, the camera center is chosen as the world coordinate. Therefore, the 3D point is expressed in the camera frame. However, it is more practical to express the coordinate of this point in the robot base frame, since we formulate all constraints and tasks in the robot base frame. As shown in Figure 4.5, the change of the frame from the center of the camera to the base frame is achieved by a homogeneous transformation $P_{cam} = \begin{bmatrix} R & t \\ 0^T & 1 \end{bmatrix} P_{base}$. Where $R \in SO(3)$ is the rotation matrix and $t \in \mathbb{R}^3$ is the cartesian shift between two frames. Thus, the relationship between a point in the robot base frame and its corresponding pixel value in the image plane is expressed by:

$$\underbrace{\begin{bmatrix} u \\ v \\ 1 \end{bmatrix}}_{\text{Pixel coordinate}} = \begin{bmatrix} \frac{f_x X}{Z} + u_0 \\ \frac{f_y Y}{Z} + v_0 \\ 1 \end{bmatrix} = \underbrace{\begin{bmatrix} f_x & 0 & u_0 & 0 \\ 0 & f_y & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}}_{\text{Intrinsic matrix}} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \underbrace{\begin{bmatrix} R & t \\ 0^T & 1 \end{bmatrix}}_{\text{Transformation matrix}} \underbrace{\begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}}_{\text{Base coordinate}} \tag{4.3}$$

The equation 4.3 gives us the projection relation between a Cartesian 3D point and the image plane. By adding the depth information from the depth image, we can therefore retrieve the corresponding keypoints pixel's Z values. This mapping requires a proper calibration of the camera [Nowak2021]. This method is also called native-lifting by [Zimmermann2018] and the results show that this method is effective regarding its simplicity and accuracy.

However, the information in the depth image is sensitive to disturbances and background features, so to make the result more robust and accurate, we define a bounding box around the keypoint to eliminate outliers and average the distance, as shown in Figure 4.3(b). Finally, we successfully map the key point's from the RGB image to the 3D location, as shown in Figure 4.3(c).

4.3 Human hand motion prediction

The dynamics of a human can be described in state-space by the Equation 4.4. Without loss of generality, we consider only the dynamics of a human's hand position to simplify the notation:

$$p_{t+1}^k = g(p_t^k, w_t^k) \tag{4.4}$$

where $p_t^k \in \mathbb{R}^3$ is the discrete time variable describing the position of the k-th vertice's of the human, $w_t^k \in \mathbb{R}^3$ is the muscle force or external effect causing the movement of the human, which is unknown, the function g represents the dynamics of the human . We assume that the human's movement is not completely random and follows patterns as shown in Figure 4.8-(b), where the dotted red lines denote representative motions to different goals.

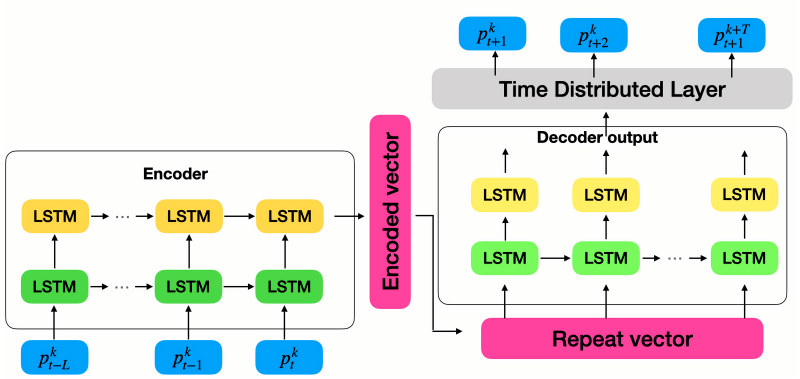


Figure 4.6: Encoder-decoder LSTMs neural network model for human motion prediction

To anticipate the human's future motion, it's not enough to predict only one-step ahead, as shown in Equation 4.4. In a more general scenario, we want to predict T steps ahead given a current state p_t^k and w_t^k and consider L-order Markov assumptions. Therefore, we can formulate this problem as: given a time-series input $p^k = \{p_t^k, p_{t-1}^k, \ldots, p_{t-L-1}^k\}$, we want to find a function ϕ such that: $\phi : p^k \rightarrow y^k$. Where L is the number of past observations and $y^k = \{p_{t+1}^k, p_{t+2}^k, \ldots, p_{t+T}^k\}$ with T being the number of steps to predict.

Modeling such a dynamic function is very challenging because the external factors are unmeasurable and unpredictable. In addition, human dynamics are highly nonlinear. However, the neural network structure is efficient to learn such nonlinear mapping patterns. We define our prediction network structure in Figure 4.6 as an encoder-decoder model. The past observation data is encoded through several stacked Long Short-Term Memory (LSTM) layers to increase the depth of the network. The LSTMs model is a special type of recurrent neural network introduced in Section 2.5. The encoded information is passed to an LSTM decoder layer, followed by a fully connected layer to produce the final multi-step prediction.

The structure of an LSTM cell is shown in Figure 4.7 and the mathematical formulation is as follows:

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \tag{4.5a}$$
$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \tag{4.5b}$$
$$\tilde{C}_t = tanh(W_C \cdot [h_t, x_t] + b_C) \tag{4.5c}$$
$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t \tag{4.5d}$$
$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t + b_o] \tag{4.5e}$$
$$h_t = o_t * tanh(Ct) \tag{4.5f}$$

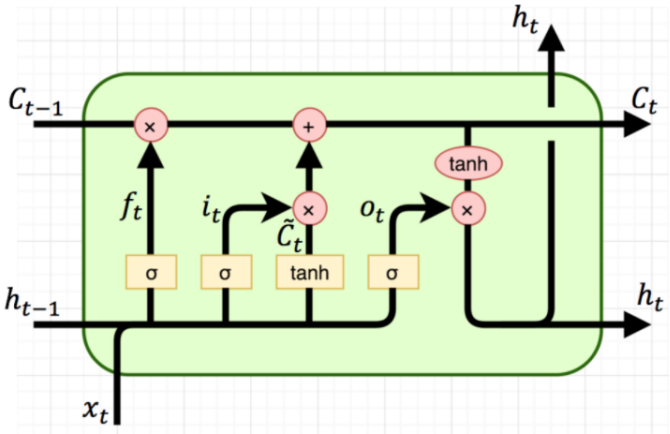


Figure 4.7: Overview of LSTM cell architecture

Where $[W_f, b_f]$, $[W_i, b_i]$, $[W_C, b_C]$ and $[W_o, b_o]$ are learnable weights and biases, f_t and i_t are forget and update gates, h_{t-1} and h_t are previous and current hidden states, respectively. f_t and i_t are the forget and update gates, respectively, and output a number between 0 and 1. $\tilde{C}_t$ is the new candidate cell value. Thus, the new cell state C_t is updated by C_{t-1} and $\tilde{C}_t$ with the associated forgetting weight and update weight. The new output and hidden states are represented by o_t and h_t.

4.3.1 Data preprocessing

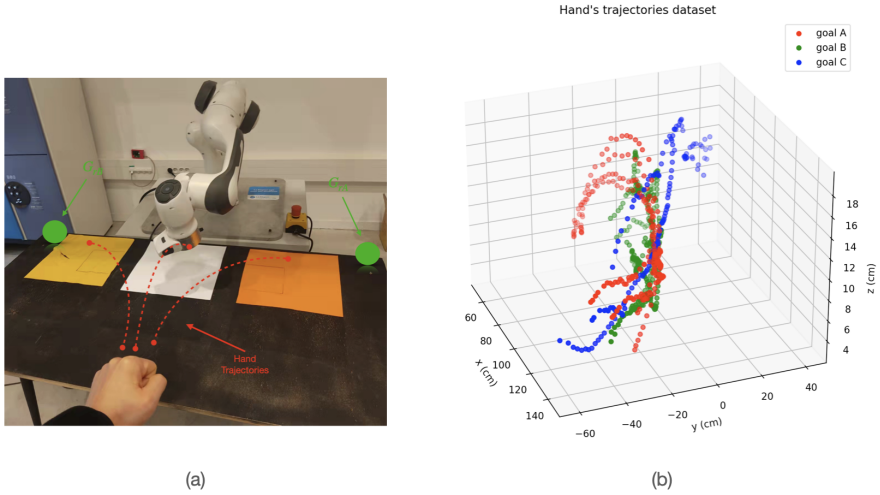


Figure 4.8: Demonstration of the environment for dataset generation, (a) shows the person working in shared environment with a collaborative robot, and shows the possible goals to which the hand should move, (b) shows a few different trajectories to reach each goal

Let us have learning data of K observation sequences collected by the RGB-D camera, S = $\{S^1, \ldots, S^K\}$, $S^k = \{S^k_1, \ldots, S^k_{T_k}\}$. Each element $S^k_{t_k}$ denotes the position of the hand in the Cartesian space obtained by the human pose detection model. Therefore, each element of a trajectory S^k is captured at the frequency of the RGB-D camera, which is at 33 Hz.

In this experiment, the human hand moves to several different goals, with some patterns shown in Figure 4.8(a) denoted by red dotted lines. For each task, we generate five similar trajectories

with small changes in position and velocity, the raw trajectory data are shown in Fig. 4.8(b). However, the raw trajectories should not be used directly in the neural network because of the noise which leads to poor training performance [Zamboni2022]. Therefore, we preprocess these trajectories to get better results in our experiments. First, we transform the absolute coordinates

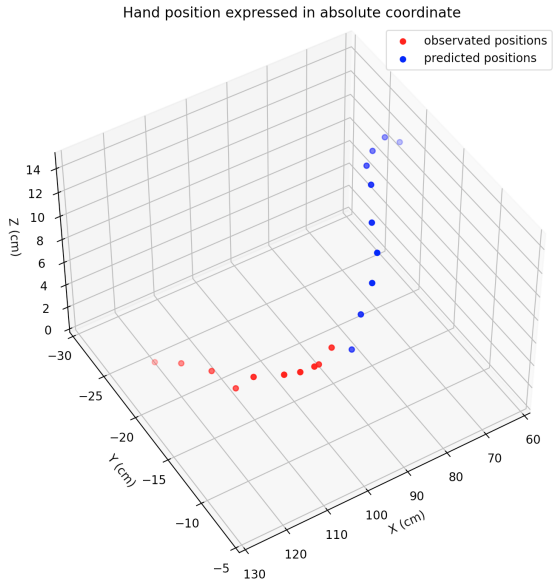


Figure 4.9: A sub-trajectory of length of 20 observations if divided into two sequences(red dots for training input and bleue dots for training output).

of the hand's position into relative coordinates (relative displacements) to make the coordinates scene independent. In this way, the neural network learns the pattern of movement displacement instead of memorizing the trajectory. Second, we apply a data augmentation method to increase the generalization ability of our neural network. We use random rotation to each trajectory to make the network learn rotation-invariant patterns. We add Gaussian noise with mean 0 and small standard deviation to each point to make the network more robust to small perturbations and inaccuracies. Further, we divide these trajectories into prediction windows. For example, we define a training sample as $\chi = (x, y)$, where x is the observed hand position for a given time-step and y is the predicted hand position, as shown in Figure 4.9, where the observations are drawn in the robot's absolute coordinate for better visualization.

4.4 Prediction results

The neural network model for hand motion prediction is shown in Figure 4.6. The encoder part is modeled by two layers of LSTMs with 64 LSTM cells per later and a small kernel regularization value to avoid overfitting. And the input data is followed by a Gaussian noise layer to increase the generalization capabilities. The decoder part is symmetric with respect to the encoder part, which is also modeled by two layers of LSTMs with 64 LSTM cells for each layer. The output of the decoder is connected to a fully connected layer to predict the relative displacement of the hand motion. The input and output time steps are both ten observations and the number of features is three, which correspond to the cartesian x,y,z coordinate of the hand. The total number of parameters of this model is equal to 116,675, which is a relatively small neural network. In

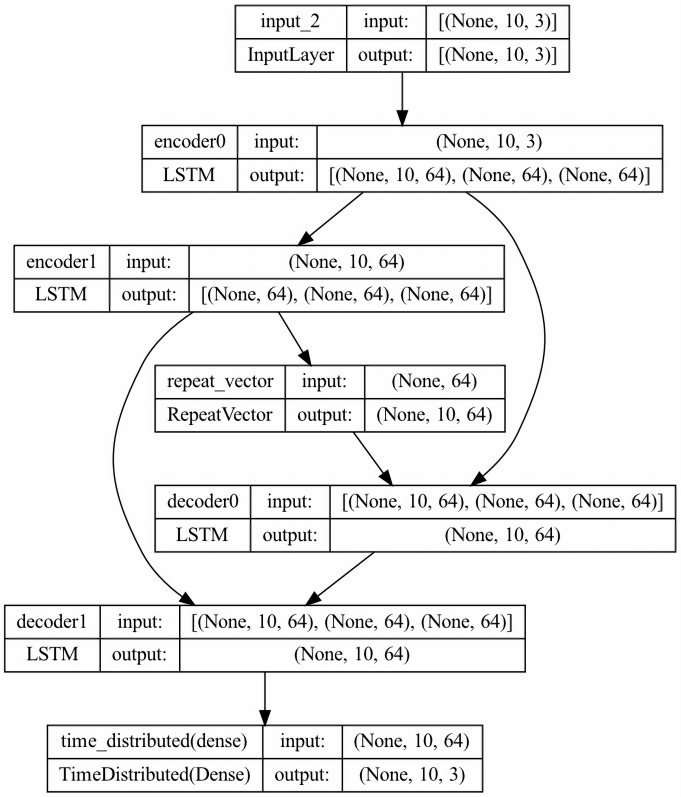


Figure 4.10: The detailed architecture of our proposed neural network model.

Figure 4.10, we give a detailed description of each layer of our neural network.

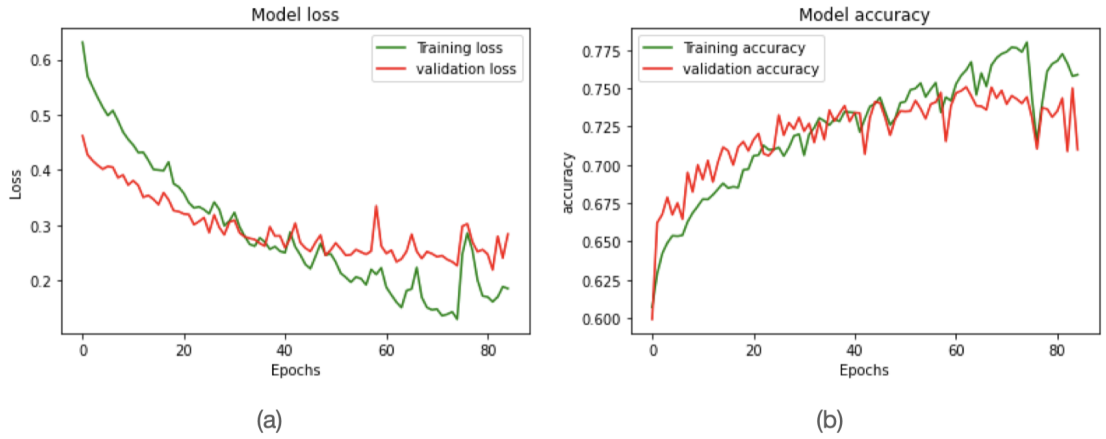


Figure 4.11: The training result of proposed model, a) show the loss function during training phase for training dataset and validation dataset, b) show the accuracy function during training phase for training dataset and validation dataset.

The training detail is shown in Figure 4.11. In Figure 4.11(a), we observe the model training and validation loss based on MSE (mean square error), which is widely used loss for regression task. In Figure 4.11(b), we have the model training and validation accuracy. The optimization algorithm for backpropagation is Adam [Kingma2014], which is an extension of stochastic gradient descent that has recently seen wider adoption for deep learning applications. The learning rate is

exponentially decaying with training steps, the initial learning rate is $1e^{-2}$, the decay rate is 0.9, and decay steps are 5000, which corresponds to

$$learning\_rate(step) = initial\_learning\_rate \times decay\_rate^{(\frac{step}{decay\_steps})} \tag{4.6}$$

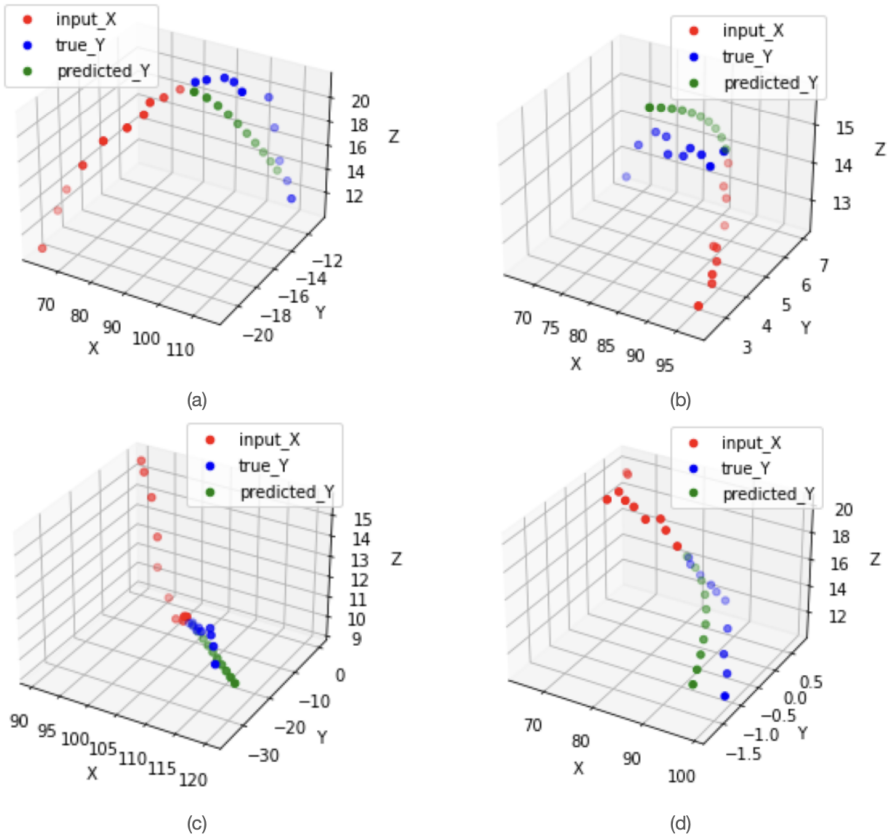


Figure 4.12: Random prediction of some dataset samples. (a) and (b) show some curved pattarns of hand movement, while (c) and (d) represent more rectilinear patterns.

In Figure 4.12, four random samples of the dataset are fed into the model (red dots), and we compare the prediction of the hand movement (green dots) with the true movement (blue dots). The model actually predicts the relative movement of the hand from the observations, and here we present the movements in absolute coordinates for a better visualization. Each input observation from these random samples shows different patterns, for example, in Figure 4.12(a) and (b), the input observations encode a curve motion, while, the input observations in Figure 4.12(c) and (d) appear to be a straight line. Then, by observing the predicted positions and the actual positions, we confirm that the model has indeed learned some patterns. If the trajectory is more of a curve, the predicted positions will also tend to be a curve, and if the trajectory is a straight line, the output will tend to be a line. Even though the results on the Figure 4.12(a), (c) and (d) are quite accurate, we still notice a clear difference between the prediction and the real positions for the sample in Figure 4.12(b). One reason is that this sample is much more noisy and represents a less common pattern than the others, yet the prediction tends to be in the right direction. This noise comes from three sources: 1) during data acquisition, 2) addition of Gaussian noise in our dataset for data augmentation, 3) Gaussian noise introduced in some neural network layers for model generalization. Surprisingly, the trajectories predicted by the neural network are very smooth, the noise is removed by the model, and a smooth trajectory is more consistent than a real movement.

Although our model has a good prediction in general, but for some observations, the prediction is very wrong compared to the real motion as shown in Figure 4.13. For example, in Figure 4.13(a),

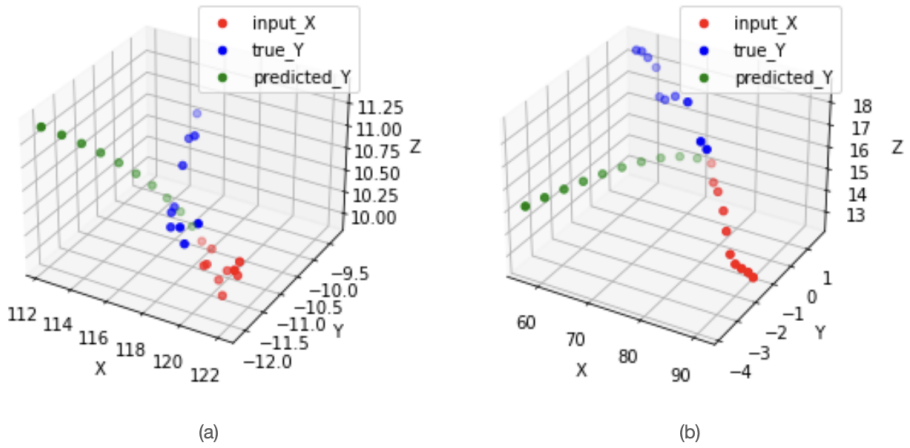


Figure 4.13: Samples for which the model does not predict the correct trajectories. a)The input observations is very noisy, b) the output trajectory gives a different pattern.

the sample seems to be wrong, because we have observations that are stationary (the ten observations are concentrated in the same small zone) and the real motion is spread over a long distance. The model does not predict the same real trajectory. And, in Figure 4.13(b), the model predicts a trajectory that is in the opposite direction of the real trajectory, this is because the hand movement is going towards three different goals and they have a part of the common trajectory as shown in Figure 4.8(a), if this common part is observed, the model prediction is ambiguous.

4.5 Conclusion

This chapter introduces the perception module to complete our general control schema 3.3. This module consists of two parts: 1) Detection of the person's pose by Mediapipe using an RGB-D camera (Asus Xtion pro). The detected pose is expressed in pixel coordinates in the image frame, then a mapping with the depth image is then applied to find the pose in Cartesian space expressed in the robot base frame. 2) Predicting the person's hand movement on a short time horizon in order to allow our trajectory generator to anticipate in the future. The hand movement is predicted by a neural network architecture called LSTM which is very effective for predicting sequential data and the prediction results on some random samples show that the model has indeed learned some hand movement patterns well.

In addition, our dataset is not perfect; the data is collected by the low-cost camera, which is highly noisy during the recording phase. In addition, some of the samples have ambiguities that erroneously predict a wrong trajectory for our model. For a better construction of the dataset, we need to use more efficient and accurate recording tools such as optiTrack.

In the next Chapter 5 we will present our software system on which we will develop the collision-free trajectory generation module, and we will see in detail how the different processing modules communicate with each other (e.g. robot state, person's hand motion, etc...). Once the different modules are established, we will experiment our trajectory generator with a real person moving next to it and check the good avoidance of potential collisions.

5

Implementation and validation

Contents

In previous chapters, we presented our control framework for collision-free trajectory generation for a manipulator robot that shares its workspace with a human. This control framework was introduced as an optimization problem in Chapter 3 which computes an optimal control input that allows the robot to reach an objective (e.g. to follow a trajectory) while respecting the imposed constraints (collision-free, joint limits, etc.). Moreover, this framework is predictive, it anticipates the future movement of the person on a short time horizon in order to have a more reactive response. Although the person's movement is quite complex, we develop a predictive model based on LSTM type neural networks to discover the movement patterns of the person's hand when it reaches an object.

In this chapter, we first describe the experimental setup and the Franka Emika control framework in Section 5.1, we present the software implementation of different modules and how they communicate data through the robot operating system framework which allows us to send command input to the robot. Then, in Section 5.2 to enable the high-level collision-free trajectory generation. The results of the experiments are presented in Section 5.3 and we conclude this chapter with the Section 5.4.

5.1 Experimentation setup

To realize and test our proposed methods, the demonstration platform shown in Figure 5.1 has been developed, where a collaborative robot manipulator works in the same space as a human operator.

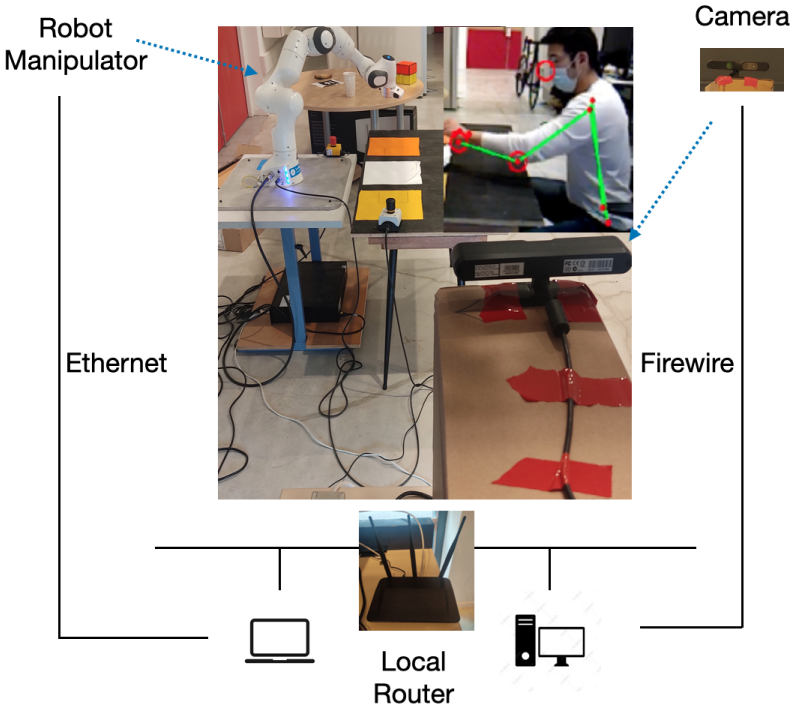


Figure 5.1: A demonstration platform for human-robot collaboration

The platform consist of several components

- A real-time framework to communicate with the robot manipulator and the trajectory gen-

erator/controller connected to an Ubuntu OS via Ethernet, ensuring online adaptation of the robot motion;

- A simulation environment for testing and debugging the proposed control framework;
- A human (hand) motion detection/prediction system using an RGB-D camera;
- Data communication via local router, Ethernet and Firewire.

The RGB-D camera is connected to the PC via Firewire, and the predicted human hand motion is then transmitted to the other PC (connected to the robot) via a local router. In the second PC, which is connected to the robot via Ethernet, the information from the perception module is used to formulate the optimization problem and solved it directly to generate a collision-free trajectory.

Since the collision-free trajectory is calculated by a high-level module, we cannot give it directly to the robot as an input command. In fact, the robot is connected to a control box that has two network ports. One is connected to the Internet and allows us to program the robot using a graphical programming interface, but this approach has limited capabilities and does not allow the user to send commands directly to the low-level controller from a high-level module. The other port can connect to a user's computer via Ethernet and execute the robot's control code using UDP (User Datagram Protocol) messages exchanged every millisecond. This is called the Franka Control Interface (FCI), which acts as an intermediary module to connect the high-level module to the low-level control input.

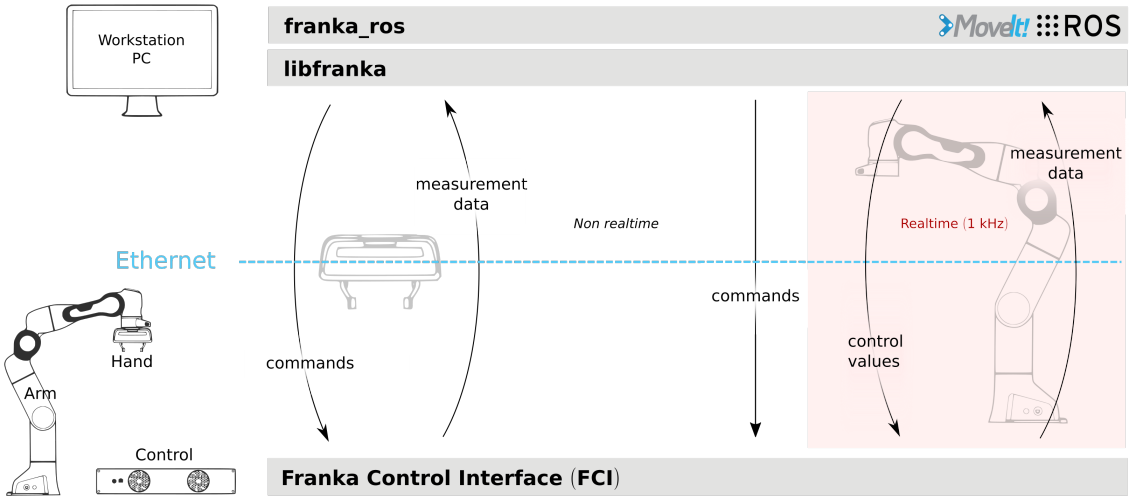


Figure 5.2: Schematic overview of Franka Control Interface.

This Franka Control Interface is accessed through a library called LibFranka and is written in C++, which allows the user's C++ code to be executed at the robot's servo frequency of 1 kHz as shown on the right side of Figure 5.2. Since FCI provides low-level bi-directional communication, the user can read the state of the robot to obtain sensor data at 1 kHz and can have the means to calculate the desired kinematic and dynamic parameters. In addition to real-time arm control, libfranka allows non real-time control with a gripper as shown in the left part of Figure 5.2.

Although LibFranka provides a fast and flexible connection to the robot arm and hand, it does not take the advantage of the benefits of open-source libraries such as the Robot Operating System (ROS). ROS is a framework for writing robot software that includes a collection of tools and libraries that simplify the task of creating complex and robust robot behavior. The philosophy of ROS is to break a large robotics project into many small programs that quickly pass messages to

each other. This paradigm facilitates and encourages the reuse of robotics software outside the specific robot and environment. Physical robots are not always be available to work with, and when they are, the process is sometimes slow. Working with ROS provides two effective workarounds to this problem: 1) Separate the low-level direct control of the hardware from the high-level processing, and we can temporarily replace these low-level programs with a simulator such as Gazebo, to test the behavior of the high-level part of the system. 2) Recording and replaying sensor data and other types of messages allow us to replay them many times to test different ways of processing the same data. In fact, there is a solution to connect LibFrank to ROS via Franka_Ros, which which allows us to explore the powerful ROS framework with Franka Emika.

In the following section, we will describe in more detail our software implementation from high-level collision-free trajectory planning to low-level robot motion control in ROS. The high-level module is consists of perception and motion generation. The low-level module is a QP optimal control package that computes optimal joint velocities that are input to the robot used by Franka_ROS.

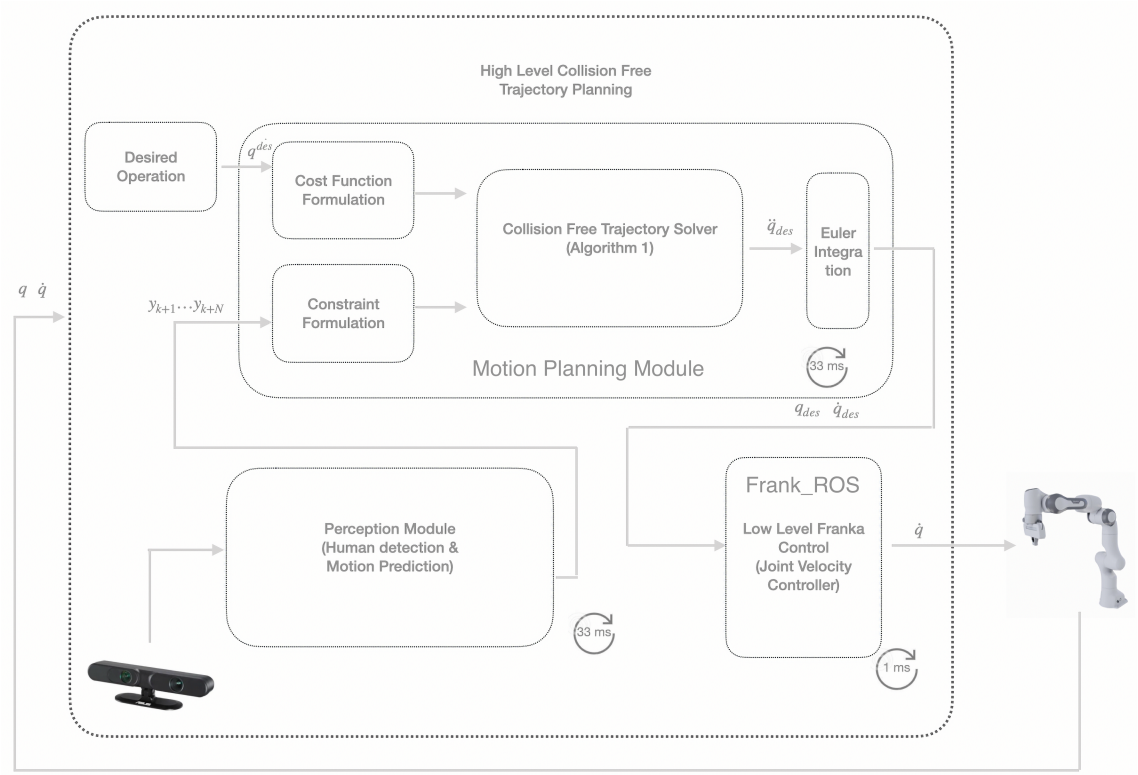


Figure 5.3: General control architecture to enable human-robot collaboration in a shared environment.

5.2 An open-source software environment based on ROS

The software architecture of our general control framework is schematized and shown in Figure 5.3, which is composed of different packages. Each package is composed of several different nodes, a node in ROS is a process that performs computation and nodes communicate with each other using streaming topics (TCP/IP-based message exchange with publish/subscribe semantics).

The camera package processes images provided by the RGB-D camera and it sends predicted hand positions to our motion planner package. The motion planner package solves online an optimization problem to generate a local collision free trajectory. This trajectory is used to control the robot via Libfranka, which is integrated into ROS by the Franka_Ros package.

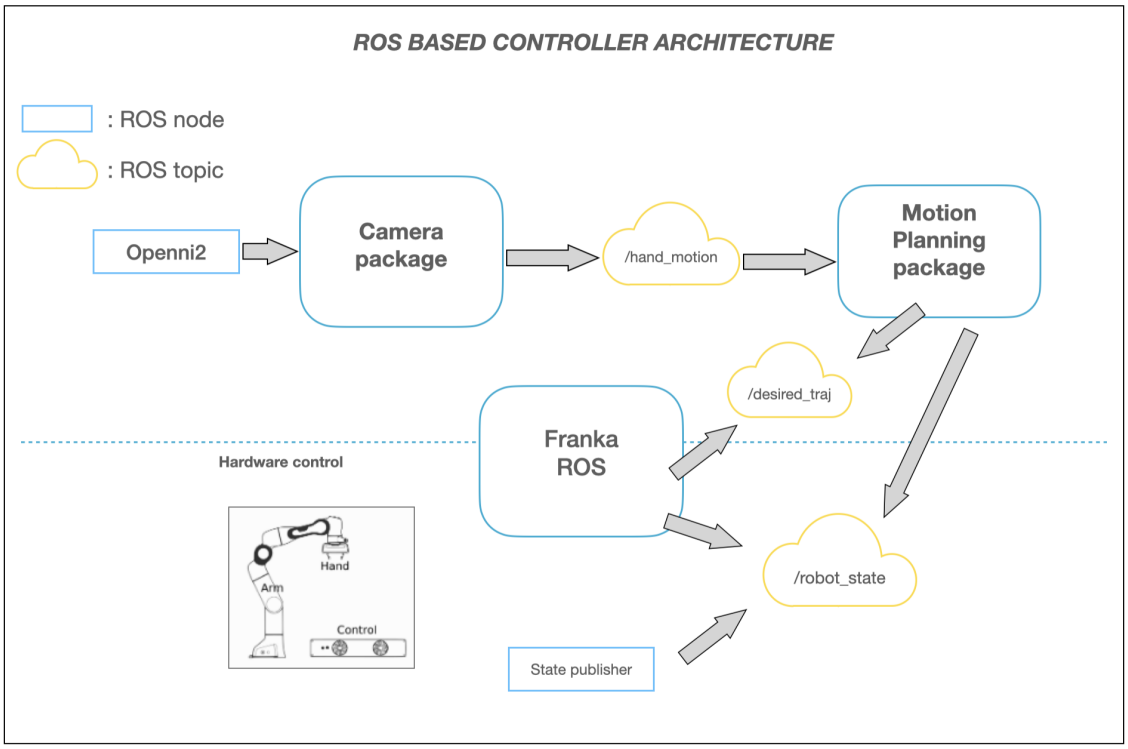


Figure 5.4: ROS implementation for the general control architecture

In Figure 5.4, we illustrate the ROS implementation of the general control architecture. A blue rectangle represents a ROS node, and a yellow cloud represents the message topic that a node can publish or subscribe to. It gives an overview of the message handling between modules, in the following subsections we will describe each module in more detail.

5.2.1 Perception package

In this work, an RGB-D camera is used to measure the robot's workspace at 30 Hz. The process described in Figure 5.5 is implemented in an open source C++/Python code. First, the openni2_launch for camera driver package is used to connect the RGB-D camera to ROS. Once the package is launched, we are able to access to different data such as color images, depth images, cloud points, etc. from the associated topic.

The neural network models presented in Chapter 4 are implemented in python using the rospy library to interface with ROS. Based on the rostopic communication facilities, the predicted hand movements of our seq2seq LSTMs model are published and later used in the motion planning package.

5.2.2 Motion planning package

Recall that our obstacle avoidance constraint is ensured by the existence of the separating planes shown in Figure 3.4. The separating_planes node subscribes to both the /hand_motion and /robot_state topics to obtain the Cartesian position of the two bodies in order to compute the separating planes

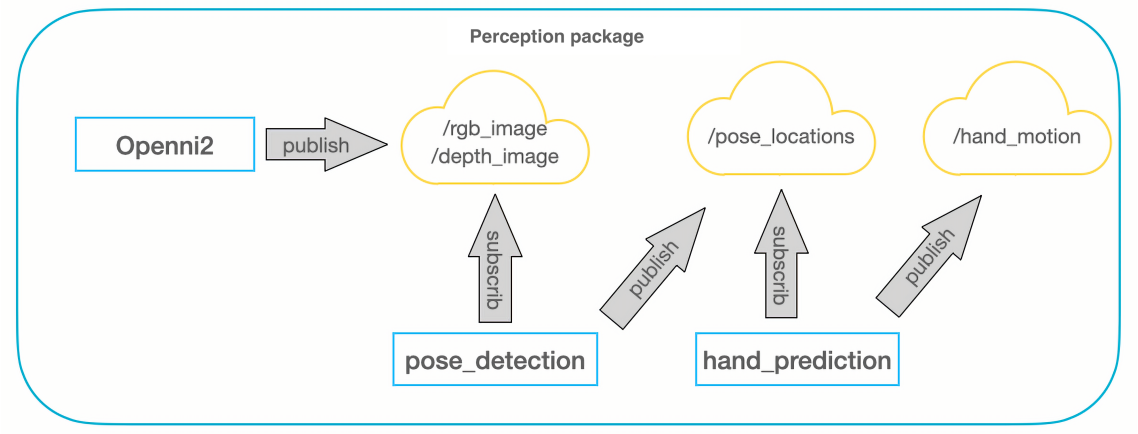


Figure 5.5: Overview of our camera package for perception module processing. The results of each processing node are communicated via rostopic.

using the qpOASES solver [Ferreau2014]. The final motion planning node computes a collision free trajectory using these planes. The desired collision-free trajectory is sent to the Franka Emika velocity controller via franka_ros package provided by the manufacturer.

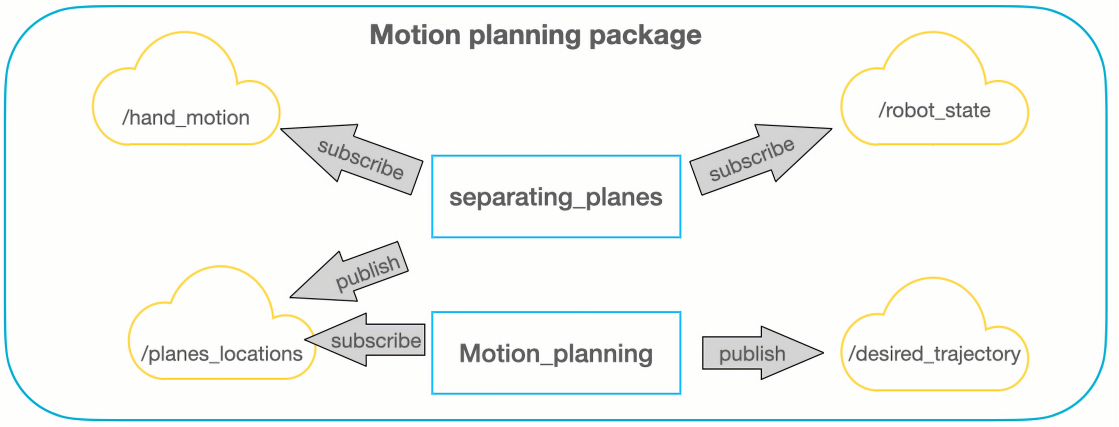


Figure 5.6: Overview of our motion planning package for collision free trajectory generation.

5.3 Application results

In this section, we present the results of the different components to validate the proposed framework. First, we test the mapping of the Mediapipe results with the depth image to validate the detection of the person's pose in Cartesian space in Section 5.3.1. Then, using a sequence of observations of the hand position, we verify the result of the hand motion prediction in Section 5.3.2. Finally, we integrate it into our MPC framework to verify the generation of collision-free trajectories in Section 5.3.3.

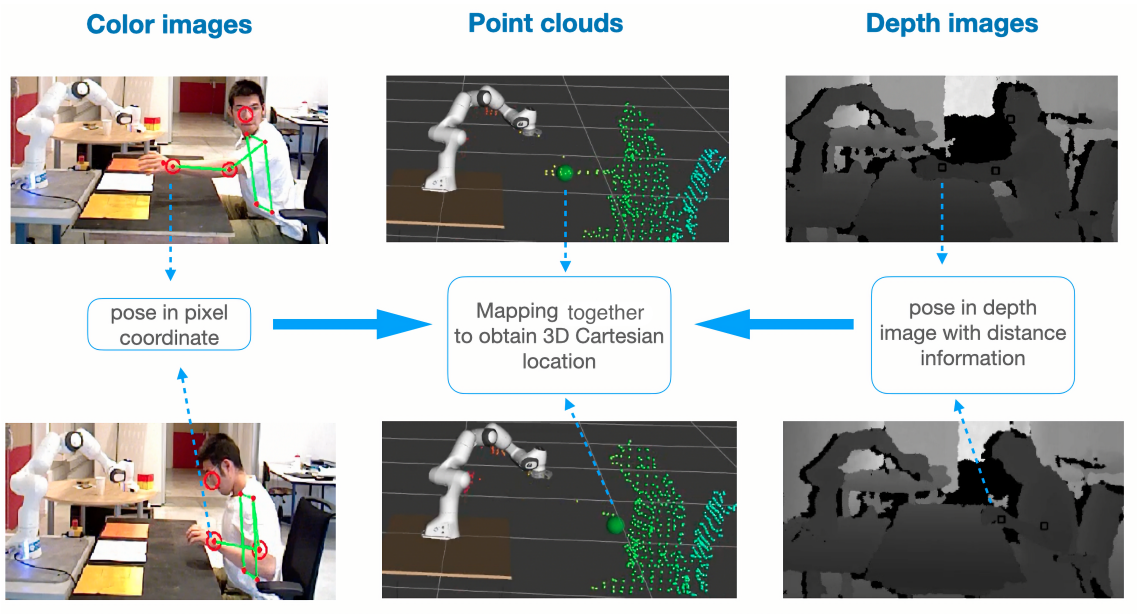


Figure 5.7: Results of the pose detection model with real image inputs. The pose location in pixel coordinates is shown in the color images (first column), the associated distance information is shown in the depth image (third column), and the final Cartesian location is obtained by mapping this information. For better visualization, the 3D position of the hand is shown in the second column in a green sphere.

5.3.1 Pose validation in Cartesian locations

In Figure 5.7, we show a person sitting in front of the robot with two different postures. The first column represents his posture in the color image, and the pose of the person is shown as a skeleton, key points such as the hand and elbow and the head are marked by red circles. These are the results obtained by Mediapipe as a pixel location in the color image reference. In the third column, we have the same scenes, but in the form of a depth image, where the depth image defines the distance of each pixel in the camera reference, and the intensity of the gray level shows the variation of the distance. Having the camera parameters after the calibration, we can use the pinhole camera model presented in Chapter 4 to reconstruct the person's position in the Cartesian space. For example, for a better visualization, in the middle column we present the position of the person in the form of point clouds. And the Cartesian position of the hand is emphasized by the green sphere.

By combining the data from the color image and the depth image with the pinhole model of the camera, we verify that we can effectively extract the position of the person's hand in Cartesian space.

5.3.2 Hand motion prediction validation

The architecture of our LSTMs model for hand motion prediction was introduced in Section 4.4. Here, with the same person sitting in front of the robot and performing a reaching task, we test the error of the prediction on the whole trajectory. In Figure 5.8, we have shown the motion of the person in the color images, and the results of the prediction of the hand's motion are shown in the first and third rows. In this test, we predict four steps ahead, and the time interval of each prediction is equal to 0.03 s, which corresponds to the camera acquisition frequency. The green sphere represents the actual observation of the hand's position, and the series of yellow spheres represents the predicted hand's motion.

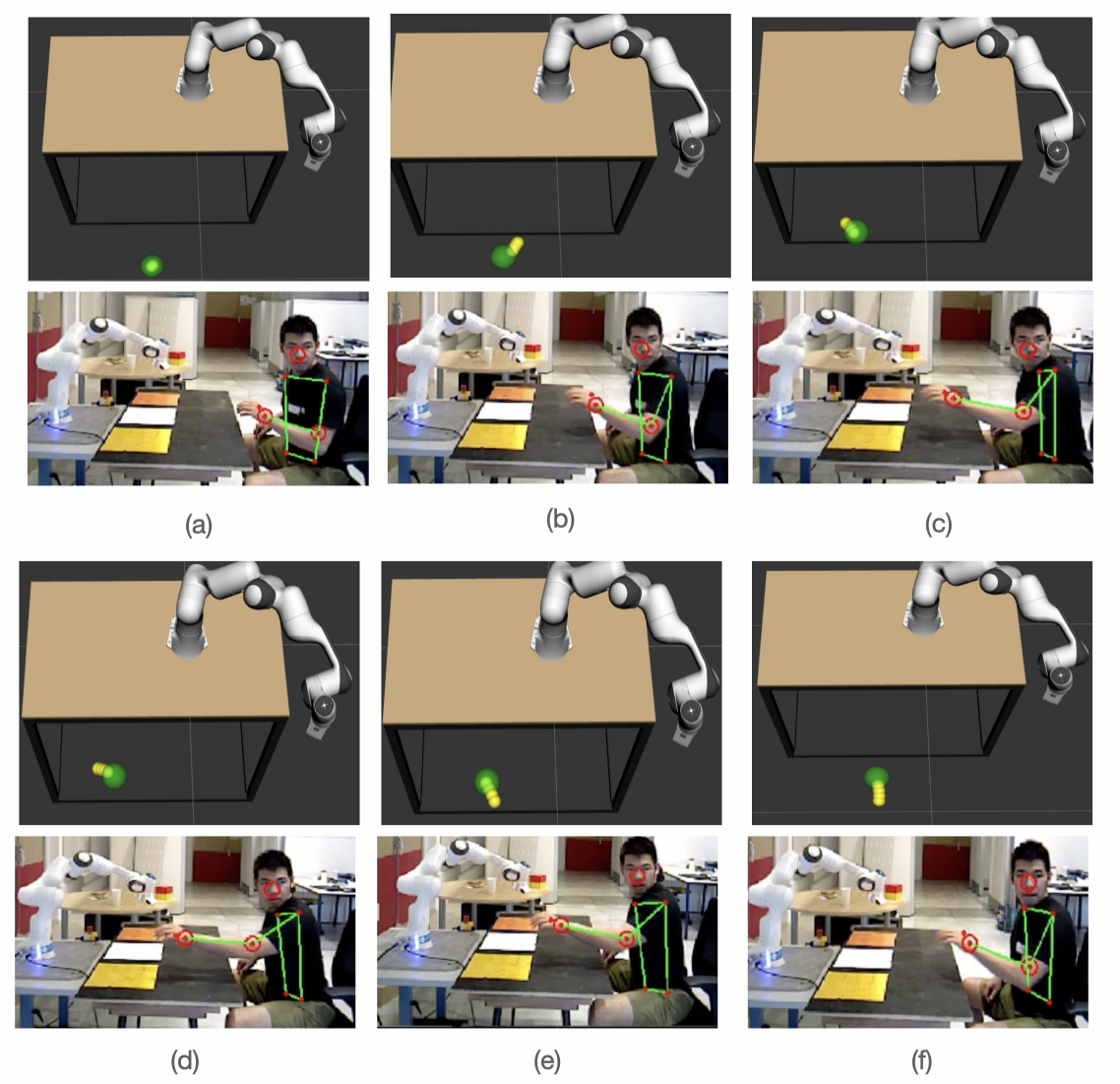


Figure 5.8: An overview of the reaching motion of the person shown in the color images. The result of the prediction of the hand's motion on a short time horizon is shown in the first and third rows. The green sphere represents the actual observation of the hand's position, and the series of yellow spheres represents the predicted hand's motion. (a) The initial position, (b)-(f) The back and forth of the reaching motion

The predicted motion compared to the true motion is shown in Figure 5.9. On the abscissa axis, we plot the mean absolute error in meters in the x, y, and z directions. On the ordinate axis, we plot the number of error samples. We can observe in the figure that most of the prediction samples have a very low error around 1 cm, however, due to the noise and imperfect dataset, sometimes the error becomes higher but remains rarely.

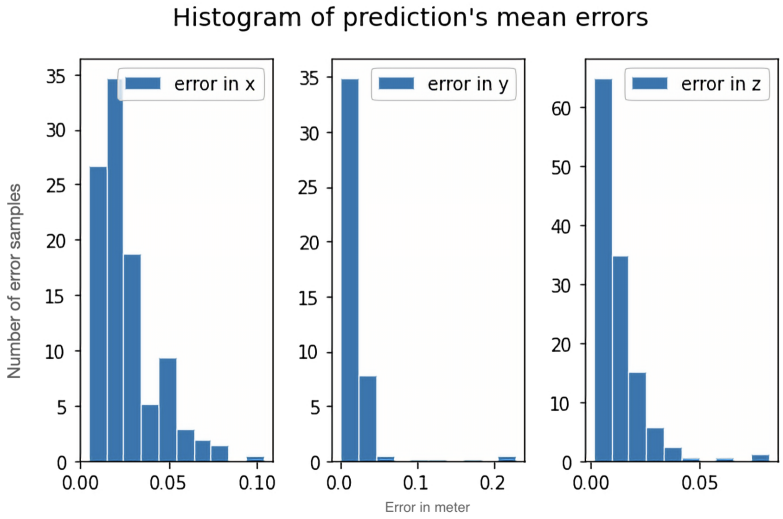


Figure 5.9: Histogram presentation of the mean prediction error

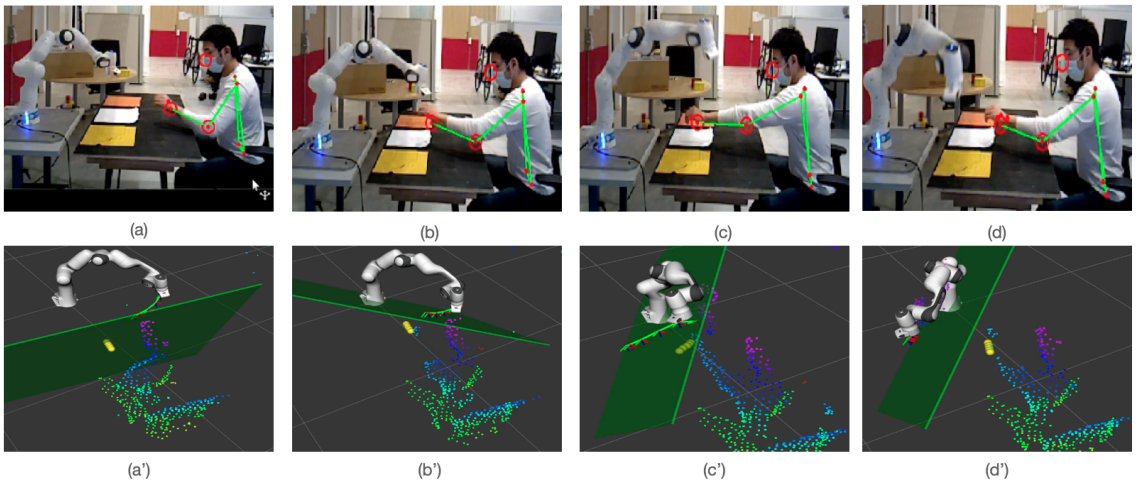


Figure 5.10: The qualitative visualization of the collaborative environment between cobot and human, a) shows the distance between cobot and hand, which is much larger than the safety distance (20cm), b) shows the behavior of the cobot when the human hand is intentionally placed for possible collision, the cobot deviates from its initial trajectory to avoid the collision, c) shows that the cobot achieves its goal without stopping or hurting the human while maintaining a safe distance. The sub-figures from (a')-(c') show the corresponding visualization of the same data in RVIZ.

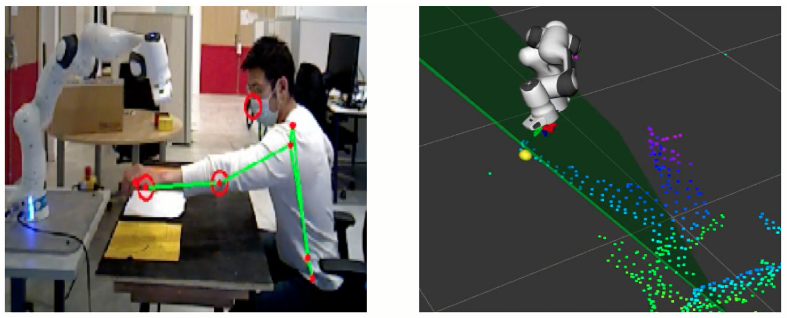


Figure 5.11: A case study whare a collision cannot be avoided, a) The human intentionally blocks the cobot's motion, b) The trajectory generator ensures that the cobot is at rest to avoid a collision.

5.3.3 Collision free trajectory generation validation

Panda's physical constraints								
Name	Joint 1	Joint 2	Joint 3	Joint 4	Joint 5	Joint 6	Joint 7	Unit
q_{max}	2.897	1.763	2.897	-0.070	2.897	3.752	2.897	rad
q_{min}	-2.897	-1.763	-2.897	-3.072	-2.897	-0.018	-2.897	rad
$\dot{q}_{max}$	2.175	2.175	2.175	2.175	2.610	2.610	2.610	$rad.s^{-1}$
$\ddot{q}_{max}$	15	7.5	10	12.5	15	20	20	$rad.s^{-2}$

Once we have validated the perception module, we integrate it into the general controller framework from ROS as shown in the Figure 5.3. The physical constraints such as joint position, joint velocity and joint acceleration are specified according to the values in the Table 5.3.3. We choose a prediction horizon of length 0.25 s, with the MPC sampling time Δt = 0.1 s and N = 5, where N is the number of steps , which covers the time necessary for the robot to stop under all circumstances, in order to satisfy the terminal constraint (3.21e) and thus enable in this way the passive motion safety guarantee. A longer prediction time could provide better collision avoidance, but this would depend heavily on the accuracy of the longer-term human motion prediction. The safety distance is chosen to be $d_{safe} = 20$ cm.

The qualitative visualization of the deployed model is shown in Figure 5.10. The safety distance is chosen equal to $d_{safe} = 20$ cm. The robot performs a pick-and-place task between the positions $G_{rA} = (0.5, 0.4, 0.2)$ m and $G_{rB} = (0.5, -0.4, 0.2)$ m expressed in the frame of the robot base link. Figure 5.10(a) and Figure 5.10(a') show the robot moving from goal G_{rA} to G_{rB}, the collision-free trajectory is shown as successive frames in green. Since the distance between the cobot and the human is large enough, this trajectory is straight to the goal. The yellow spheres represent the predicted positions of the human hand in five time-steps. The green plane represents the separating plane (only the first predicted step is shown here, we have a total of (N-1) planes). In Figure 5.10(b-c) and Figure 5.10(b'-c'), the cobot deviates its trajectory in order to avoid the human motion. The predicted positions are shown by the yellow sphere. Finally, the cobot visits the position of G_{rB} with successful collision avoidance, as shown in Figure 5.10(d) and Figure 5.10(d').

To ensure human safety in collaborative shared workspace, we evaluated different cases where we ensured that cobot achieves its goal by avoiding the possible collision with human, as explained in previous experiment, and we also tested a situation which creates a deadlock and it is not achievable for the cobot to complete the task. The Figure 5.11 shows the result where the human hand is placed for longer time, which does not creates an exception to complete the task. In this

case, the motion generator keeps the cobot still at the desired safe distance.

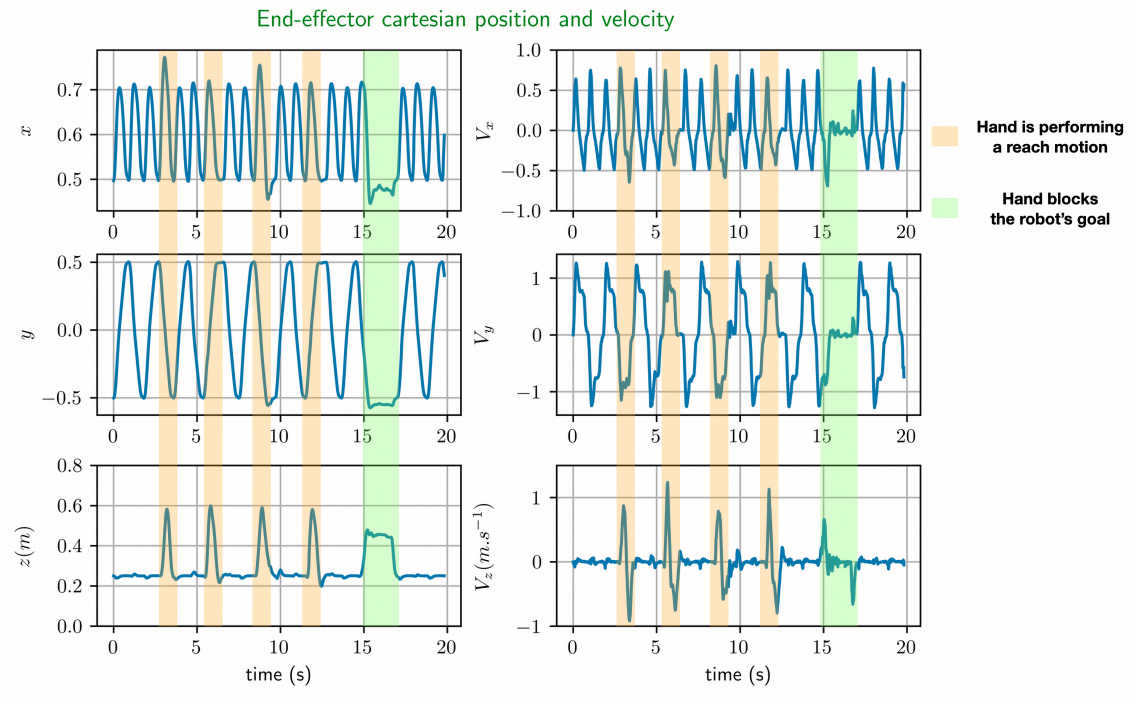


Figure 5.12: The variation of robot's end-effector Cartesian position and its linear velocity.

To measure the robot's performance in avoiding of the person's hand movement. We show in the Figure 5.12 the variation of the robot's end-effector position and its linear velocity. When the person's hand interferes with the robot's initial trajectory, the robot changes its height (we observe an increase in its z coordinate) to avoid the potential collision with the hand. However, the performance of the robot is not much affected by this perturbation, because we observe that the velocities in the x and y coordinates remain almost identical, despite of the change in the velocity of the z coordinate. If there is no perturbation with the movement of motion the hand, the velocity in z direction remains zero.

In Figure 5.13, we compare two trajectories generated with different prediction horizons (N=2 and N=5) to illustrate the advantage of anticipating the future motions. When N=2, the trajectory generator produces only two steps of motion, and this prediction horizon is not able to predict in advance the motion of the person on its future trajectory, so the robot starts to avoid the person's hand lately, only that they are very close (it can violate the safety distance) as shown in the first two rows of Figure 5.13(a-c). As long as for N=5, the trajectory generator predicts an interference between the hand motion and the previous trajectory of the robot, so it starts to plan a new trajectory to avoid the collision as shown in the last two rows of Figure 5.13(a-c).

5.4 Conclusion

In this chapter, we have introduced the software architecture of our MPC based motion generator. We have explained in detail the processes of the different modules (e.g. perception module and motion planning module) and the data communication between these modules. We then test and validate these modules through experiments in a real-world scenario. These experiments show that

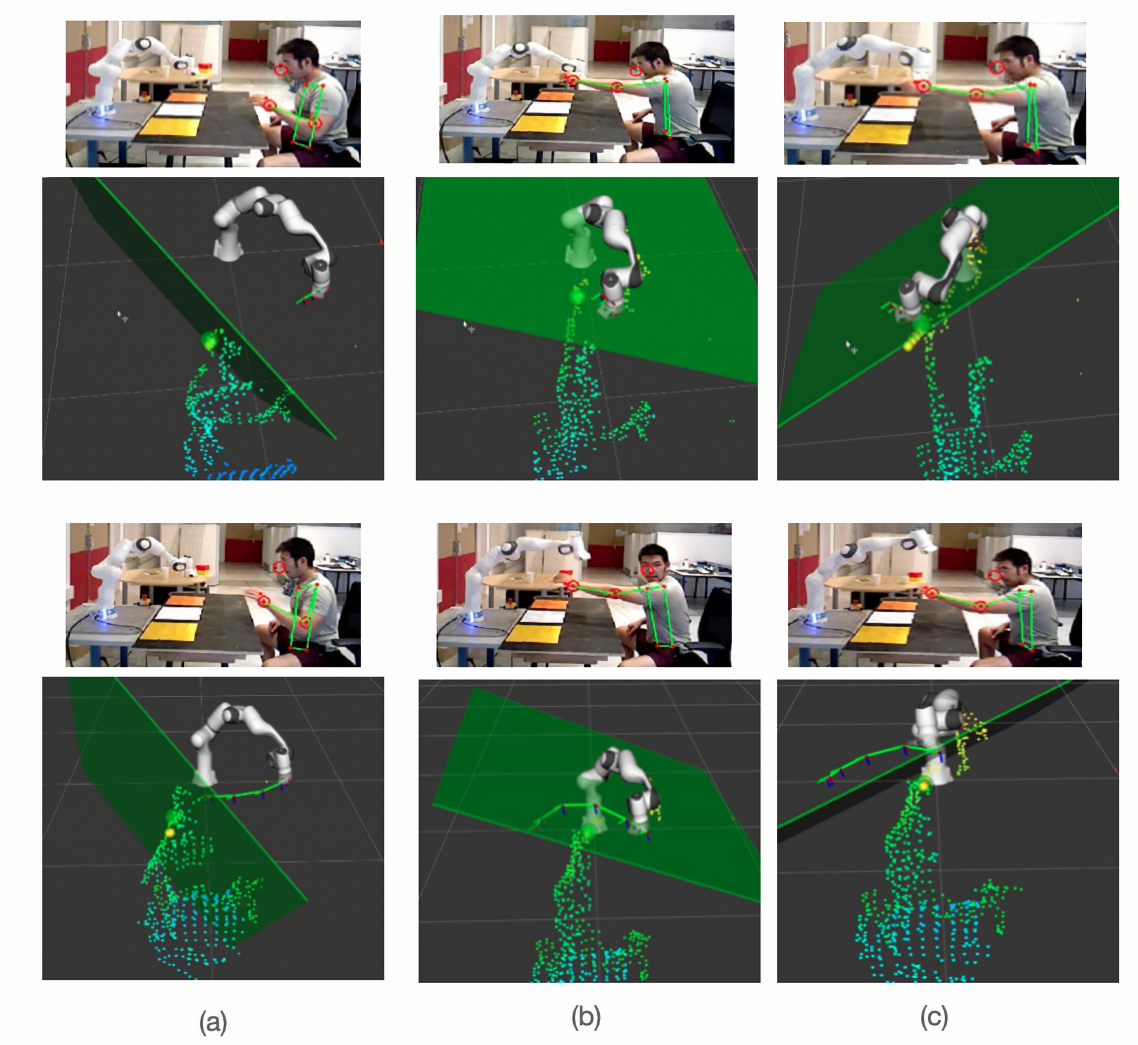


Figure 5.13: Illustration of the impact of the prediction horizon for the generated trajectory when the hand moves at a much faster speed.

even when the initial trajectory of the robot is perturbed by the movement of the person's hand, the tasks are performed correctly, and the input control satisfies the constraints. Moreover, the MPC predictive framework allows to anticipate the movement of the person present in the workspace in order to avoid in advance a possible collision with the imposed safety distance. If the prediction horizon is too short or not predicted at all, the robot will act too late when the person's hand approaches, which may results in the safety distance not being maintained or even have a collision. We conclude that MPC approaches show the desired behavior; however they still require more work in the implementation and need more experiments for different scenarios.

6 Conclusion

Contents

The work presented in this thesis opens new interesting directions in the field of programming collaborative robots in a shared workspace with humans. Several questions are raised in this context:

1. How to select and design a framework for generating collision-free trajectories in real-time, anticipating future changes in the environment ? How to properly consider the closest distance between the robot and the human ?
2. How to predict the future movement of the person presented in the shared workspace ?
3. How to implement this framework in the terms of software ?

6.1 Summary

In this thesis, we have presented a novel framework to deal with real-time collision avoidance for robots performing tasks in a shared workspace. We have developed a reactive trajectory generation algorithm that takes into account the future motion of the environment, following an MPC design, and provides a practical way to generate locally collision-free optimal solutions. The solutions are summarized in detail in this section.

6.1.1 Online motion generation in a shared workspace

The framework for generating collision-free trajectories is the Model Predictive Control (MPC) framework, which not only improves the reactivity of the system, but also provides a more accurate local linear approximations of the collision avoidance constraints. The passively-safety is ensured by an appropriate choice of the terminal constraint. The generated trajectories are defined in joint space because this allows easy integration of several constraints (e.g. joint range, joint velocity, joint acceleration, etc ...).

The collision avoidance constraint in Cartesian space can be easily transported to joint space by Jacobian and linearization techniques. This ensures that the closest distance between robot and human is continuously differentiable, we adopt the idea of separating planes, which ensures that two objects don't collide over two intervals of time step by finding the existence of a plane.

We also transform our nonlinear motion generation optimization problem into two quadratic programming (QP) problems because solving a nonlinear problem with a nonlinear solver is quite slow. We first compute a given set of plane positions by solving a first QP with the robot joint acceleration constant, then use the set of planes to solve the second QP with fixed planes. Solving the first QP again gives new values of planes positions to compute the next iteration and repeat this process until convergence.

6.1.2 Human motion prediction

Predicting the human's motion is essential for generating a reactive collision-free trajectory, in this thesis, we detect the human's upper-body keypoints, but only predict the hand's motion over a short time horizon. The scenarios studied involve a closed distance collaboration between a robot manipulator and a worker which both are performing some pick-and-place tasks. Thus, the relevant motion of the person is the hand's motion, which needs to extract its associated motion patterns.

The motion prediction is divided into two parts: 1) Detection of the person's pose by Mediapipe using an RGB-D camera (Asus Xtion pro). The detected pose is expressed in pixel coordinates in the image frame, then a mapping with the depth image is then applied to find the pose in

Cartesian space expressed in the robot base frame. 2) Predicting the hand movement of the person on a short time horizon in order to allow our trajectory generator to anticipate in the future. The hand movement is predicted by a neural network architecture called LSTM, which is very effective for predicting sequential data and the prediction results on some random samples show that the model has indeed learned some hand movement patterns well.

6.1.3 Implementation solution

Online computability is a necessary and a key component to validate our framework. The software implementations are realized through several different components: 1) a real-time framework to communicate with the robot manipulator and the trajectory generation module, 2) a simulation environment for testing and debugging, 3) a human (hand) motion detection/prediction module, 4) data communication strategies.

The low-level communication with robot is realized by using the libFranka package written in C++, which allows us to send commands to control Franka Emika. The higher level communication between hand motion prediction and trajectory generation was implemented in ROS, they comminucate using streaming topic (TCP/IP based message exchange with publish/subsribe semantics) to allow formulating QP optimization problem.

The proposed approach has been successfully realized in the demonstration platform. The result described in Chapter 5 show that the robot can complete its tasks without much decreasing its performing while avoidance collisions.

6.2 Future work

The concepts introduced in this thesis raise several questions for future work.

- Throughout this thesis, the robot trajectories have been defined at the kinematic level, knowing that there is a low level tracking controller that translates kinematic variables into motor commands (libFranka). The dynamics of the robot and its hardware constraints are not explicitly considered. However, they are implicity taken into account through various constraints (e.g. joint range, velocity and acceleration). An interesting improvement to this work is to define robot trajectories at the dynamic level and apply the real physical constraints of the robot by explicitly considering the hardware constraints of the robot, such as motor torques;

- In Chapter 5, we have validated our formulations by implementing them for online trajectory generation for collaborative robots in a shared workspace. However, the scenario has been greatly simplified, we only provide the motion of the hand. It could be interesting to include the whole body or upper body motion prediction of the human, which will bring the possibilities to integrate more complex collaborative task. In addition, the nonlinear optimization have been approximated by iteratively solving two QP, which only provide a suboptimal solution, an important future work will be to improve nonlinear optimization numerical tools;

- In Chapter 4, the hand motion dataset were collected from a single RGB-D camera, which is noisy and imprecise. During this thesis, there is no publicly available human robot close collaboration dataset. A long-term future work would be to build a well designed human robot collaboration dataset that includes several different scenarios with a motion capture system. This will be an interesting development that will bring new opportunities to all

collaboratif robotic communicty to implement and test there algorithms and open the door to human robot collaboration;

- The motion prediction model outputs a set of deterministic future positions, but human motion is very complex and cannot be accurately predict, which can lead to risky decisions for robots. A future direction of work will be to integrate the probabilistic framework into motion prediction, which will provide safer and higher quality trajectories, because it can incorporate uncertainty that tells the robot if the outputs are reliable;
- In this thesis, we developed a reactive trajectory controller that plans over a short time horizon following an MPC design, and it provides a locally optimal solutions. Another future work would be to integrate our current approach with a global planning algorithm so that we end up with a reactive trajectory that works in real time and generates a global time optimal solution. solution.

Bibliography

[Albu-Schäffer2007] Alin Albu-Schäffer, Sami Haddadin, Ch Ott, Andreas Stemmer, Thomas Wimböck et Gerhard Hirzinger. *The DLR lightweight robot: design and control concepts for robots in human environments*. Industrial Robot: an international journal, 2007.

[Amato1996] Nancy M Amato et Yan Wu. *A randomized roadmap method for path and manipulation planning*. In Proceedings of IEEE international conference on robotics and automation, volume 1, pages 113–120. IEEE, 1996.

[Ames2016] Aaron D Ames, Xiangru Xu, Jessy W Grizzle et Paulo Tabuada. *Control barrier function based quadratic programs for safety critical systems*. IEEE Transactions on Automatic Control, vol. 62, no. 8, pages 3861–3876, 2016.

[Balan2006] Lucian Balan et Gary M Bone. *Real-time 3D collision avoidance method for safe human and robot coexistence*. In 2006 IEEE/RSJ international conference on intelligent robots and systems, pages 276–282. IEEE, 2006.

[Bejarano2019] Ronal Bejarano, Borja Ramis Ferrer, Wael M Mohammed et Jose L Martinez Lastra. *Implementing a Human-Robot Collaborative Assembly Workstation*. In 2019 IEEE 17th International Conference on Industrial Informatics (INDIN), volume 1, pages 557–564. IEEE, 2019.

[Bhattacharya2008] Priyadarshi Bhattacharya et Marina L Gavrilova. *Roadmap-based path planning-using the voronoi diagram for a clearance-based shortest path*. IEEE Robotics & Automation Magazine, vol. 15, no. 2, pages 58–66, 2008.

[Bishop2006] Christopher M Bishop et Nasser M Nasrabadi. Pattern recognition and machine learning, volume 4. Springer, 2006.

[Bouraine2012] Sara Bouraine, Thierry Fraichard et Hassen Salhi. *Provably safe navigation for mobile robots with limited field-of-views in dynamic environments*. Autonomous Robots, vol. 32, no. 3, pages 267–283, 2012.

[Boyd2004a] Stephen Boyd, Stephen P Boyd et Lieven Vandenberghe. Convex optimization. Cambridge university press, 2004.

[Boyd2004b] Stephen Boyd, Stephen P Boyd et Lieven Vandenberghe. Convex optimization. Cambridge university press, 2004.

[Brossette2017] Stanislas Brossette et Pierre-Brice Wieber. *Collision avoidance based on separating planes for feet trajectory generation*. In 2017 IEEE-RAS

17th International Conference on Humanoid Robotics (Humanoids), pages 509–514. IEEE, 2017.

[Caldera2018] Shehan Caldera, Alexander Rassau et Douglas Chai. *Review of deep learning methods in robotic grasp detection*. Multimodal Technologies and Interaction, vol. 2, no. 3, page 57, 2018.

[Cao2017] Zhe Cao, Tomas Simon, Shih-En Wei et Yaser Sheikh. *Realtime multi-person 2d pose estimation using part affinity fields*. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 7291–7299, 2017.

[Choset2005] Howie Choset, Kevin M Lynch, Seth Hutchinson, George A Kantor et Wolfram Burgard. Principles of robot motion: theory, algorithms, and implementations. MIT press, 2005.

[Connolly1990] Christopher I Connolly, J Brian Burns et Rich Weiss. *Path planning using Laplace's equation*. In Proceedings., IEEE International Conference on Robotics and Automation, pages 2102–2106. IEEE, 1990.

[Ding2011a] Hao Ding, Gunther Reißig, Kurniawan Wijaya, Dino Bortot, Klaus Bengler et Olaf Stursberg. *Human arm motion modeling and long-term prediction for safe and efficient human-robot-interaction*. In 2011 IEEE International Conference on Robotics and Automation, pages 5875–5880. IEEE, 2011.

[Ding2011b] Hao Ding, Gunther Reißig, Kurniawan Wijaya, Dino Bortot, Klaus Bengler et Olaf Stursberg. *Human arm motion modeling and long-term prediction for safe and efficient human-robot-interaction*. In 2011 IEEE International Conference on Robotics and Automation, pages 5875–5880. IEEE, 2011.

[Ding2011c] Hao Ding, Gunther Reißig, Kurniawan Wijaya, Dino Bortot, Klaus Bengler et Olaf Stursberg. *Human arm motion modeling and long-term prediction for safe and efficient human-robot-interaction*. In 2011 IEEE International Conference on Robotics and Automation, pages 5875–5880. IEEE, 2011.

[Djuric2016] Ana M Djuric, RJ Urbanic et JL Rickli. *A framework for collaborative robot (CoBot) integration in advanced manufacturing systems*. SAE International Journal of Materials and Manufacturing, vol. 9, no. 2, pages 457–464, 2016.

[Docekal2022] Jan Docekal, Jakub Rozlivek, Jiri Matas et Matej Hoffmann. *Human keypoint detection for close proximity human-robot interaction*. arXiv preprint arXiv:2207.07742, 2022.

[Duchoň2014] František Duchoň, Andrej Babinec, Martin Kajan, Peter Beňo, Martin Florek, Tomáš Fico et Ladislav Jurišica. *Path planning with modified a star algorithm for a mobile robot*. Procedia Engineering, vol. 96, pages 59–69, 2014.

[Emika] Franka Emika. *Panda*. Rapport technique, Accessed.[Online]. Available: https://www.franka.de.

[Fabrizio2016] Flacco Fabrizio et Alessandro De Luca. *Real-time computation of distance to dynamic obstacles with multiple depth sensors*. IEEE Robotics and Automation Letters, vol. 2, no. 1, pages 56–63, 2016.

[Faverjon1987] Bernard Faverjon et Pierre Tournassoud. *A local based approach for path planning of manipulators with a high number of degrees of freedom*. In Proceedings. 1987 IEEE international conference on robotics and automation, volume 4, pages 1152–1159. IEEE, 1987.

[Ferreau2014] Hans Joachim Ferreau, Christian Kirches, Andreas Potschka, Hans Georg Bock et Moritz Diehl. *qpOASES: A parametric active-set algorithm for quadratic programming*. Mathematical Programming Computation, vol. 6, no. 4, pages 327–363, 2014.

[Flacco2012a] Fabrizio Flacco, Torsten Kröger, Alessandro De Luca et Oussama Khatib. *A depth space approach to human-robot collision avoidance*. In 2012 IEEE international conference on robotics and automation, pages 338–345. IEEE, 2012.

[Flacco2012b] Fabrizio Flacco, Torsten Kröger, Alessandro De Luca et Oussama Khatib. *A depth space approach to human-robot collision avoidance*. In 2012 IEEE international conference on robotics and automation, pages 338–345. IEEE, 2012.

[Flash1985] Tamar Flash et Neville Hogan. *The coordination of arm movements: an experimentally confirmed mathematical model*. Journal of neuroscience, vol. 5, no. 7, pages 1688–1703, 1985.

[for Standardization2011] International Organization for Standardization. *Iso 10218-1: 2011 robots and robotic devices—safety requirements for industrial robots—part 1: Robots*. International Organization for Standardization, 2011.

[Fraichard2004] Thierry Fraichard et Hajime Asama. *Inevitable collision states—a step towards safer robots?* Advanced Robotics, vol. 18, no. 10, pages 1001–1024, 2004.

[Fridovich-Keil2020] David Fridovich-Keil, Andrea Bajcsy, Jaime F Fisac, Sylvia L Herbert, Steven Wang, Anca D Dragan et Claire J Tomlin. *Confidence-aware motion prediction for real-time collision avoidance1*. The International Journal of Robotics Research, vol. 39, no. 2-3, pages 250–265, 2020.

[Fujii2016] Masakazu Fujii, Hiroki Murakami et Mitsuharu Sonehara. *Study on application of a human-robot collaborative system using hand-guiding in a production line*. IHI Eng. Rev, vol. 49, no. 1, pages 24–29, 2016.

[Gao2020] Qing Gao, Jinguo Liu et Zhaojie Ju. *Robust real-time hand detection and localization for space human–robot interaction based on deep learning*. Neurocomputing, vol. 390, pages 198–206, 2020.

[Garrido2010] Santiago Garrido, Luis Moreno, Dolores Blanco et Fernando Martín Monar. *Robotic motion using harmonic functions and finite elements*. Journal of intelligent and Robotic Systems, vol. 59, no. 1, pages 57–73, 2010.

[Gilbert1988] Elmer G Gilbert, Daniel W Johnson et S Sathiya Keerthi. *A fast procedure for computing the distance between complex objects in three-dimensional space*. IEEE Journal on Robotics and Automation, vol. 4, no. 2, pages 193–203, 1988.

[Grujić2015a] Tamara Grujić et Mirjana Bonković. *Measurement and analysis of human hand kinematics*. World Academy of Science, Engineering and Technology International Journal of Biomedical and Biological Engineering, vol. 9, no. 2, page 97, 2015.

[Grujic2015b] Tamara Grujic et Mirjana Bonkovic. *Measurement and analysis of human hand kinematics*. International Journal of Medical, Health, Biomedical and Pharmaceutical Engineering, vol. 9, pages 97–102, 2015.

[Haddadin2008] Sami Haddadin, Alin Albu-Schaffer, Alessandro De Luca et Gerd Hirzinger. *Collision detection and reaction: A contribution to safe physical human-robot interaction*. In 2008 IEEE/RSJ International Conference on Intelligent Robots and Systems, pages 3356–3363. IEEE, 2008.

[Haque2016] Albert Haque, Boya Peng, Zelun Luo, Alexandre Alahi, Serena Yeung et Li Fei-Fei. *Towards viewpoint invariant 3d human pose estimation*. In European conference on computer vision, pages 160–177. Springer, 2016.

[Hsu2006] David Hsu, Jean-Claude Latombe et Hanna Kurniawati. *On the probabilistic foundations of probabilistic roadmap planning*. The International Journal of Robotics Research, vol. 25, no. 7, pages 627–643, 2006.

[Huang2004] Han-Pang Huang et Shu-Yun Chung. *Dynamic visibility graph for path planning*. In 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566), volume 3, pages 2813–2818. IEEE, 2004.

[Hwang1992] Yong Koo Hwang, Narendra Ahuja*et al. A potential field approach to path planning.* IEEE Transactions on Robotics and Automation, vol. 8, no. 1, pages 23–32, 1992.

[Jiang1987] Bernard C Jiang et Charles A Gainer Jr. *A cause-and-effect analysis of robot accidents*. Journal of Occupational accidents, vol. 9, no. 1, pages 27–45, 1987.

[Joseph2018] Lucas Joseph, Vincent Padois et Guillaume Morel. *Towards X-ray medical imaging with robots in the open: safety without compromising performances*. In 2018 IEEE International Conference on Robotics and Automation (ICRA), pages 6604–6610. IEEE, 2018.

[Kakadiaris1996] Ioannis A Kakadiaris et Dimitris Metaxas. *Model-based estimation of 3D human motion with occlusion based on active multi-viewpoint selection*. In Proceedings CVPR IEEE Computer Society Conference on Computer Vision and Pattern Recognition, pages 81–87. IEEE, 1996.

[Kanazawa2018] Angjoo Kanazawa, Michael J Black, David W Jacobs et Jitendra Malik. *End-to-end recovery of human shape and pose*. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 7122–7131, 2018.

[Kanehiro2008] Fumio Kanehiro, Florent Lamiraux, Oussama Kanoun, Eiichi Yoshida et Jean-Paul Laumond. *A local collision avoidance method for non-strictly convex polyhedra*. Proceedings of robotics: science and systems IV, page 33, 2008.

[Kavraki1996] Lydia E Kavraki, Petr Svestka, J-C Latombe et Mark H Overmars. *Probabilistic roadmaps for path planning in high-dimensional configuration spaces*. IEEE transactions on Robotics and Automation, vol. 12, no. 4, pages 566–580, 1996.

[Khatib1986] Oussama Khatib. *Real-time obstacle avoidance for manipulators and mobile robots*. In Autonomous robot vehicles, pages 396–404. Springer, 1986.

[Kim1992] Jin-Oh Kim et Pradeep Khosla. *Real-time obstacle avoidance using harmonic potential functions*. 1992.

[Kimm2015a] Dennis Kimm et David V Thiel. *Hand speed measurements in boxing*. Procedia Engineering, vol. 112, pages 502–506, 2015.

[Kimm2015b] Dennis Kimm et David V Thiel. *Hand speed measurements in boxing*. Procedia Engineering, vol. 112, pages 502–506, 2015.

[Kimmel2017] Melanie Kimmel et Sandra Hirche. *Invariance control for safe human–robot interaction in dynamic environments*. IEEE Transactions on Robotics, vol. 33, no. 6, pages 1327–1342, 2017.

[Kingma2014] Diederik P Kingma et Jimmy Ba. *Adam: A method for stochastic optimization*. arXiv preprint arXiv:1412.6980, 2014.

[Kratzer2019] Philipp Kratzer, Marc Toussaint et Jim Mainprice. *Motion prediction with recurrent neural network dynamical models and trajectory optimization*. arXiv preprint arXiv:1906.12279, 2019.

[Kuffner2000] James J Kuffner et Steven M LaValle. *RRT-connect: An efficient approach to single-query path planning*. In Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065), volume 2, pages 995–1001. IEEE, 2000.

[Lasota2015] Przemyslaw A Lasota et Julie A Shah. *Analyzing the effects of human-aware motion planning on close-proximity human–robot collaboration*. Human factors, vol. 57, no. 1, pages 21–33, 2015.

[LaValle2006] Steven M LaValle. Planning algorithms. Cambridge university press, 2006.

[Lee2003] Min Cheol Lee et Min Gyu Park. *Artificial potential field based path planning for mobile robots using a virtual obstacle concept*. In Proceedings 2003 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM 2003), volume 2, pages 735–740. IEEE, 2003.

[Li2014] Sijin Li et Antoni B Chan. *3d human pose estimation from monocular images with deep convolutional neural network*. In Asian Conference on Computer Vision, pages 332–347. Springer, 2014.

[Lingelbach2004] Frank Lingelbach. *Path planning using probabilistic cell decomposition*. In IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA'04. 2004, volume 1, pages 467–472. IEEE, 2004.

[Liu2019] Zitong Liu, Quan Liu, Wenjun Xu, Zhihao Liu, Zude Zhou et Jie Chen. *Deep learning-based human motion prediction considering context awareness for human-robot collaboration in manufacturing*. Procedia CIRP, vol. 83, pages 272–278, 2019.

[Lowe2004] G Lowe. *Sift-the scale invariant feature transform*. Int. J, vol. 2, no. 91-110, page 2, 2004.

[Lynch2017] Kevin M Lynch et Frank C Park. Modern robotics. Cambridge University Press, 2017.

[Madarash-Hill2004] Cherie Madarash-Hill et JB Hill. *Enhancing access to ieee conference proceedings: a case study in the application of ieee xplore full text and table of contents enhancements*. Science & Technology Libraries, vol. 24, no. 3-4, pages 389–399, 2004.

[Magrini2020] Emanuele Magrini, Federica Ferraguti, Andrea Jacopo Ronga, Fabio Pini, Alessandro De Luca et Francesco Leali. *Human-robot coexistence and interaction in open industrial cells*. Robotics and Computer-Integrated Manufacturing, vol. 61, page 101846, 2020.

[Mainprice2013a] Jim Mainprice et Dmitry Berenson. *Human-robot collaborative manipulation planning using early prediction of human motion*. In 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems, pages 299–306. IEEE, 2013.

[Mainprice2013b] Jim Mainprice et Dmitry Berenson. *Human-robot collaborative manipulation planning using early prediction of human motion*. In 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems, pages 299–306. IEEE, 2013.

[Mainprice2015] Jim Mainprice, Rafi Hayne et Dmitry Berenson. *Predicting human reaching motion in collaborative tasks using inverse optimal control and iterative re-planning*. In 2015 IEEE International Conference on Robotics and Automation (ICRA), pages 885–892. IEEE, 2015.

[Martinez2017a] Julieta Martinez, Michael J Black et Javier Romero. *On human motion prediction using recurrent neural networks*. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 2891–2900, 2017.

[Martinez2017b] Julieta Martinez, Rayat Hossain, Javier Romero et James J Little. *A simple yet effective baseline for 3d human pose estimation*. In Proceedings of the IEEE international conference on computer vision, pages 2640–2649, 2017.

[Matúšová2019] Miriam Matúšová, Marcela Bučányová et Erika Hrušková. *The future of industry with collaborative robots*. In MATEC Web of Conferences, volume 299, page 02008. EDP Sciences, 2019.

[Mayne2000] David Q Mayne, James B Rawlings, Christopher V Rao et Pierre OM Scokaert. *Constrained model predictive control: Stability and optimality*. Automatica, vol. 36, no. 6, pages 789–814, 2000.

[Meguenani2015] Anis Meguenani, Vincent Padois et Philippe Bidaud. *Control of robots sharing their workspace with humans: an energetic approach to safety*. In 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 4678–4684. IEEE, 2015.

[Mehrpouya2019] Mehrshad Mehrpouya, Amir Dehghanghadikolaei, Behzad Fotovvati, Alireza Vosooghnia, Sattar S Emamian et Annamaria Gisario. *The potential of additive manufacturing in the smart factory industrial 4.0: A review*. Applied Sciences, vol. 9, no. 18, page 3865, 2019.

[Moreno-Noguer2017] Francesc Moreno-Noguer. *3d human pose estimation from a single image via distance matrix regression*. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 2823–2832, 2017.

[Murray2017] Richard M Murray, Zexiang Li et S Shankar Sastry. A mathematical introduction to robotic manipulation. CRC press, 2017.

[Murty1988] Katta G Murty et Feng-Tien Yu. Linear complementarity, linear and nonlinear programming, volume 3. Citeseer, 1988.

[Nowak2021] Jordan Nowak, Philippe Fraisse, Andrea Cherubini et Jean-Pierre Daures. *Point Clouds With Color: A Simple Open Library for Matching RGB and Depth Pixels from an Uncalibrated Stereo Pair*. In 2021 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI), pages 1–7. IEEE, 2021.

[Ogura2012] Yu Ogura, Masakazu Fujii, Kazuyuki Nishijima, Hiroki Murakami et Mitsuharu Sonehara. *Applicability of hand-guided robot for assembly-line work*. Journal of Robotics and Mechatronics, vol. 24, no. 3, pages 547–552, 2012.

[Pereira2017a] Aaron Pereira et Matthias Althoff. *Overapproximative human arm occupancy prediction for collision avoidance*. IEEE Transactions on Automation Science and Engineering, vol. 15, no. 2, pages 818–831, 2017.

[Pereira2017b] Aaron Pereira et Matthias Althoff. *Overapproximative human arm occupancy prediction for collision avoidance*. IEEE Transactions on Automation Science and Engineering, vol. 15, no. 2, pages 818–831, 2017.

[Pierson2017] Harry A Pierson et Michael S Gashler. *Deep learning in robotics: a review of recent research*. Advanced Robotics, vol. 31, no. 16, pages 821–835, 2017.

[Polverini2014] Matteo Parigi Polverini, Andrea Maria Zanchettin et Paolo Rocco. *Real-time collision avoidance in human-robot interaction based on kinetostatic safety field*. In 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems, pages 4136–4141. IEEE, 2014.

[Quinlan1993] Sean Quinlan et Oussama Khatib. *Elastic bands: Connecting path planning and control*. In [1993] Proceedings IEEE International Conference on Robotics and Automation, pages 802–807. IEEE, 1993.

[Ren2006] Jing Ren, Kenneth A McIsaac et Rajnikant V Patel. *Modified Newton's method applied to potential field-based navigation for mobile robots*. IEEE Transactions on Robotics, vol. 22, no. 2, pages 384–391, 2006.

[Rudenko2020] Andrey Rudenko, Luigi Palmieri, Michael Herman, Kris M Kitani, Dariu M Gavrila et Kai O Arras. *Human motion trajectory prediction: A survey*. The International Journal of Robotics Research, vol. 39, no. 8, pages 895–935, 2020.

[Sarbolandi2015] Hamed Sarbolandi, Damien Lefloch et Andreas Kolb. *Kinect range sensing: Structured-light versus Time-of-Flight Kinect*. Computer vision and image understanding, vol. 139, pages 1–20, 2015.

[Schulman2014] John Schulman, Yan Duan, Jonathan Ho, Alex Lee, Ibrahim Awwal, Henry Bradlow, Jia Pan, Sachin Patil, Ken Goldberg et Pieter Abbeel. *Motion planning with sequential convex optimization and convex collision checking*. The International Journal of Robotics Research, vol. 33, no. 9, pages 1251–1270, 2014.

[Scott2004] Stephen H Scott. *Optimal feedback control and the neural basis of volitional motor control*. Nature Reviews Neuroscience, vol. 5, no. 7, pages 532–545, 2004.

[Shotton2012] Jamie Shotton, Ross Girshick, Andrew Fitzgibbon, Toby Sharp, Mat Cook, Mark Finocchio, Richard Moore, Pushmeet Kohli, Antonio Criminisi, Alex Kipman*et al. Efficient human pose estimation from single depth images*. IEEE transactions on pattern analysis and machine intelligence, vol. 35, no. 12, pages 2821–2840, 2012.

[Shu2011] Chang Shu, Xiaoqing Ding et Chi Fang. *Histogram of the oriented gradient for face recognition*. Tsinghua Science and Technology, vol. 16, no. 2, pages 216–224, 2011.

[Spong2006] Mark W Spong, Seth Hutchinson, Mathukumalli Vidyasagar*et al.* Robot modeling and control, volume 3. Wiley New York, 2006.

[Staudemeyer2019] Ralf C Staudemeyer et Eric Rothstein Morris. *Understanding LSTM–a tutorial into long short-term memory recurrent neural networks*. arXiv preprint arXiv:1909.09586, 2019.

[Sylla2014a] Nahema Sylla, Vincent Bonnet, Gentiane Venture, Nahid Armande et Philippe Fraisse. *Human arm optimal motion analysis in industrial screwing task*. In 5th IEEE RAS/EMBS international conference on biomedical robotics and biomechatronics, pages 964–969. IEEE, 2014.

[Sylla2014b] Nahema Sylla, Vincent Bonnet, Gentiane Venture, Nasser Armande et Philippe Fraisse. *Human arm optimal motion analysis in industrial screwing task*. In 5th IEEE RAS/EMBS International Conference on Biomedical Robotics and Biomechatronics, pages 964–969. IEEE, 2014.

[Thrun2002] Sebastian Thrun. *Probabilistic robotics*. Communications of the ACM, vol. 45, no. 3, pages 52–57, 2002.

[Toyer2017] Sam Toyer, Anoop Cherian, Tengda Han et Stephen Gould. *Human pose forecasting via deep markov models*. In 2017 International Conference on Digital Image Computing: Techniques and Applications (DICTA), pages 1–8. IEEE, 2017.

[Tsai2014] Chi-Shen Tsai, Jwu-Sheng Hu et Masayoshi Tomizuka. *Ensuring safety in human-robot coexistence environment*. In 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems, pages 4191–4196. IEEE, 2014.

[Unhelkar2014] Vaibhav V Unhelkar, Jorge Perez, James C Boerkoel, Johannes Bix, Stefan Bartscher et Julie A Shah. *Towards control and sensing for an autonomous mobile robotic assistant navigating assembly lines*. In 2014 IEEE International Conference on Robotics and Automation (ICRA), pages 4161–4167. IEEE, 2014.

[Uno1989] Yoji Uno, Mitsuo Kawato et Rika Suzuki. *Formation and control of optimal trajectory in human multijoint arm movement*. Biological cybernetics, vol. 61, no. 2, pages 89–101, 1989.

[Vysocky2016] ALES Vysocky et PETR Novak. *Human-Robot collaboration in industry*. MM Science Journal, vol. 9, no. 2, pages 903–906, 2016.

[Wang2017] Yiwei Wang, Xin Ye, Yezhou Yang et Wenlong Zhang. *Collision-free trajectory planning in human-robot interaction through hand movement prediction from vision*. In 2017 IEEE-RAS 17th International Conference on Humanoid Robotics (Humanoids), pages 305–310. IEEE, 2017.

[Wang2018] Peng Wang, Hongyi Liu, Lihui Wang et Robert X Gao. *Deep learning-based human motion recognition for predictive context-aware human-robot collaboration*. CIRP annals, vol. 67, no. 1, pages 17–20, 2018.

[Wang2020] Kangkan Wang, Jin Xie, Guofeng Zhang, Lei Liu et Jian Yang. *Sequential 3D human pose and shape estimation from point clouds*. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 7275–7284, 2020.

[Wang2021] Jinbao Wang, Shujie Tan, Xiantong Zhen, Shuo Xu, Feng Zheng, Zhenyu He et Ling Shao. *Deep 3D human pose estimation: A review*. Computer Vision and Image Understanding, vol. 210, page 103225, 2021.

[Wright1997] Stephen J Wright. Primal-dual interior-point methods. SIAM, 1997.

[Xu2020] Tianxu Xu, Dong An, Zhonghan Wang, Sicheng Jiang, Chengnuo Meng, Yiwen Zhang, Qiang Wang, Zhongqi Pan et Yang Yue. *3D Joints Estimation of the Human Body in Single-Frame Point Cloud*. IEEE Access, vol. 8, pages 178900–178908, 2020.

[Xu2021] Tianxu Xu, Dong An, Yuetong Jia et Yang Yue. *A review: Point cloud-based 3d human joints estimation*. Sensors, vol. 21, no. 5, page 1684, 2021.

[Ye2014] Mao Ye et Ruigang Yang. *Real-time simultaneous pose and shape estimation for articulated objects using a single depth camera*. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 2345–2352, 2014.

[Yuan2017] Xiaohui Yuan, Longbo Kong, Dengchao Feng et Zhenchun Wei. *Automatic feature point detection and tracking of human actions in time-of-flight videos*. IEEE/CAA Journal of Automatica Sinica, vol. 4, no. 4, pages 677–685, 2017.

[Zamboni2022] Simone Zamboni, Zekarias Tilahun Kefato, Sarunas Girdzijauskas, Christoffer Norén et Laura Dal Col. *Pedestrian trajectory prediction with convolutional neural networks*. Pattern Recognition, vol. 121, page 108252, 2022.

[Zeiler2014] Matthew D Zeiler et Rob Fergus. *Visualizing and understanding convolutional networks*. In European conference on computer vision, pages 818–833. Springer, 2014.

[Zhang2020a] Fan Zhang, Valentin Bazarevsky, Andrey Vakunov, Andrei Tkachenka, George Sung, Chuo-Ling Chang et Matthias Grundmann. *Mediapipe hands: On-device real-time hand tracking*. arXiv preprint arXiv:2006.10214, 2020.

[Zhang2020b] Jianjing Zhang, Hongyi Liu, Qing Chang, Lihui Wang et Robert X Gao. *Recurrent neural network for motion trajectory prediction in human-robot collaborative assembly*. CIRP annals, vol. 69, no. 1, pages 9–12, 2020.

[Zheng2020] Pu Zheng, Pierre-Brice Wieber et Olivier Aycard. *Online optimal motion generation with guaranteed safety in shared workspace*. In 2020 IEEE International Conference on Robotics and Automation (ICRA), pages 9210–9215. IEEE, 2020.

[Zhou2016] Xingyi Zhou, Xiao Sun, Wei Zhang, Shuang Liang et Yichen Wei. *Deep kinematic pose regression*. In European Conference on Computer Vision, pages 186–201. Springer, 2016.

[Zhou2020] Yufan Zhou, Haiwei Dong et Abdulmotaleb El Saddik. *Learning to estimate 3d human pose from point cloud*. IEEE Sensors Journal, vol. 20, no. 20, pages 12334–12342, 2020.

[Zhu2010] Youding Zhu, Behzad Dariush et Kikuo Fujimura. *Kinematic self retargeting: A framework for human pose estimation*. Computer vision and image understanding, vol. 114, no. 12, pages 1362–1375, 2010.

[Zimmermann2018] Christian Zimmermann, Tim Welschehold, Christian Dornhege, Wolfram Burgard et Thomas Brox. *3d human pose estimation in rgbd images for robotic task learning*. In 2018 IEEE International Conference on Robotics and Automation (ICRA), pages 1986–1992. IEEE, 2018.

Vers des mouvements sûrs pour un bras robotique opérant à proximité d'humains

Towards safe robot arm motion close to humans.

Résumé

Les robots collaboratifs offrent de nouvelles possibilités d'utilisation des robots dans les espaces de travail partagés avec les humains. Ces robots peuvent interagir avec leur environnement et assister les êtres humains dans leurs tâches de manière plus sûre que les robots industriels standard. Ils doivent être rapides, précis et efficaces dans l'accomplissement de leurs tâches. Cependant, leur force peut en faire des outils dangereux pour les personnes. Par conséquent, pour assurer la sécurité, ils sont souvent utilisés de manière sous-optimale.
L'objectif de ce travail est d'assurer la sécurité d'un humain interagissant avec un robot effectuant un ensemble de tâches. Il est plus particulièrement axé sur la génération de trajectoires sans collision. Le robot doit être capable d'anticiper le mouvement de l'homme pour pouvoir initier des actions à temps et rendre possible la planification de trajectoires sans collision à long terme. La méthode proposée est validée en l'implémentant pour la génération de trajectoire en ligne sur un manipulateur robotique série de 7 dof. Ce robot est utilisé dans un espace de travail partagé avec un humain. En utilisant des capteurs externes, il est démontré qu'il est possible de réaliser des tâches tout en évitant de manière réactive une collision potentielle.

Mots-clés : Collaboration Homme-Robot, Commande Prédictive, Sécurité, Programmation Quadratique, Robots Redondants, Prédiction du Mouvement, Apprentissage Profond, Planification du Mouvement.

Abstract

Collaborative robots offer new possibilities to use robots in workspaces shared with humans. These robots can interact with their environment and assist human beings in their task in a safer way compared to standard industrial ones. They are required to be fast, precise, and efficient during the accomplishment of their tasks. However, their strength can make them dangerous tools around people. Therefore, to ensure safety they are often used in a sub-optimal way.
The aim of this work is to ensure the safety of a human interacting with a robot performing a set of tasks. It is more specifically focused on the collision-free trajectory generation. The robot needs to be able to anticipate human motion to be able to initiate actions in time and to render long–term collision-free trajectory planning possible.The proposed method is validated by implementing them for online trajectory generation on a 7 dof serial robotic manipulator. This robot is used in a shared workspace with a human. Using external sensors it is shown that it is possible to realise tasks while reactively avoiding a potential collision.

Keywords : Human/Robot Collaboration, Model Predictive Control, Safety, Quadratic Programming, Redundant Robots, Motion Prediction, Deep Learning, Motion Planning