

THÈSE

En vue de l'obtention du

DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE

Délivré par : *l'Université Toulouse 3 Paul Sabatier (UT3 Paul Sabatier)*

Présentée et soutenue le *31/01/2023* **par :**

Jérôme TRUC

Human-aware motion planning and control for a flying coworker

---

JURY

Francis COLAS	Chargé de Recherche	Examinateur
Philippe FRAISSE	Professeur	Examinateur
Jacques GANGLOFF	Professeur	Examinateur
Simon LACROIX	Directeur de Recherche	Président du Jury
Florent LAMIRAUX	Directeur de Recherche	Examinateur
Jessica CAUCHARD	Professeure Associée	Rapportrice
Cédric PRADALIER	Professeur Associé	Rapporteur
Daniel SIDOBRE	Maître de Conférences	Directeur de Thèse
Rachid ALAMI	Directeur de Recherche Emérite	Invité
Serena IVALDI	Chargée de Recherche	Invitée

École doctorale et spécialité :
EDSYS : Robotique 4200046
Unité de Recherche :
LAAS-CNRS
Directeur de Thèse :
Daniel SIDOBRE
Rapporteurs :
Jessica CAUCHARD et *Cédric PRADALIER*

Acknowledgments

Je souhaite en premier lieu remercier les membres du jury : à commencer par Jessica CAUCHARD et Cédric PRADALIER pour leur rôle de rapporteur/rapportrice de mes travaux de thèse, le président du jury Simon LACROIX ainsi que les examinateurs Francis COLAS, Philippe FRAISSE, Jacques GANGLOFF et Florent LAMIRAUX. Merci également aux invités Rachid ALAMI et Serena IVALDI pour leur intérêt. Je suis ravi que vous ayez apprécié la présentation de mes travaux lors de la soutenance. J'ai apprécié également la diversité de nos échanges, les nombreux membres du jury spécialisés dans des domaines divers ont soulevé des questions très intéressantes.

Je remercie chaleureusement mon directeur de thèse Daniel SIDOBRE pour son suivi assidu de mes travaux en toute circonstance. Toujours disponible, il a su être à l'écoute et m'a apporté tous les outils nécessaires à la réalisation de mes travaux notamment au travers du développement d'une extension de la librairie SoftMotion. Au delà du travail de thèse, il y a la vie personnelle, avec des hauts et des bas et je ne remercierai jamais assez Daniel d'en avoir tenu compte avec toute son humanité.

Merci également à Rachid ALAMI pour son suivi de près de mes travaux et ses précieux conseils notamment sur les aspects intéraction humain robot. Nos échanges ont toujours été enrichissants et productifs, m'ayant permis d'apporter ma pierre à l'édifice dans le vaste domaine de la robotique sur un projet complexe.

Je n'oublie pas le laboratoire du LAAS-CNRS, qui m'a acceuilli avec tout le confort nécessaire à la réalisation de mes travaux. Merci de même à l'équipe RIS et tous les membres qui la composent. Toujours agréables et disponibles pour aider, ils contribuent au confort et à la qualité du travail effectué dans le laboratoire.

Je remercie mes parents pour m'avoir fourni tout ce dont j'avais besoin pour en arriver là où j'en suis. Ils ont toujours répondu présents quelque soit mes choix personnels ou professionnels.

Merci à mes amis de m'avoir permis de relativiser et décompresser durant ces moments intenses.

Enfin je remercie infiniment ma compagne Marion pour son soutien sans faille au quotidien et pour avoir su gérer mon caractère. Je la remercie également pour le bonheur apporté par la naissance de notre fille Julia, née au cours de cette thèse dont elle s'est occupée admirablement durant cette phase où je n'étais parfois pas toujours disponible.

Contents

1 Introduction **1**
- 1.1 Context . . . 1
- 1.2 Flying CoWorker project . . . 2
- 1.3 Publications . . . 6
  - 1.3.1 Published . . . 6
  - 1.3.2 Submitted . . . 6
- 1.4 Manuscript organization . . . 7

2 Human-aware reactive navigation planning for a Flying CoWorker **9**
- 2.1 Introduction . . . 9
- 2.2 Related work . . . 10
- 2.3 KHAOS . . . 12
  - 2.3.1 Initial path . . . 13
  - 2.3.2 Human-aware Costs . . . 15
  - 2.3.3 Kinematic constraints . . . 17
  - 2.3.4 KHAOS main algorithm . . . 17
  - 2.3.5 Local and global trajectory costs . . . 18
  - 2.3.6 Convergence criterion . . . 20
- 2.4 Results . . . 21
  - 2.4.1 Simulation environment . . . 22
  - 2.4.2 Examples . . . 23
- 2.5 Discussion and future work . . . 34
- 2.6 Conclusion . . . 35

3 KHAOS improvements Using BSpline **37**
- 3.1 Introduction . . . 37
- 3.2 Issues . . . 38
- 3.3 Tests of existing solutions . . . 40
- 3.4 KHAOS improvements using B-Spline . . . 46
  - 3.4.1 Introduction to B-Spline . . . 47
  - 3.4.2 SoftMotion extension using Non-Uniform Cubic B-Spline . . . 49
- 3.5 Results . . . 51
  - 3.5.1 Initial path smoothing . . . 51
  - 3.5.2 Kinematic compliance . . . 52
- 3.6 Conclusion . . . 55

4 Human-FCW Handover planning and control **57**
- 4.1 Introduction . . . 57
- 4.2 Flying CoWorker design and handover . . . 58
- 4.3 Related work . . . 60

4.4 Flying CoWorker base orientation for Handover 62
4.4.1 Orientation calculation details 63
4.4.2 Orientation behavior during approach 63
4.4.3 Discussion 65
4.5 Reactive FCW goal state estimation for handover 66
4.5.1 Context 66
4.5.2 Reactive goal estimation 68
4.6 Coordinated motion for FCW 79
4.6.1 Context 80
4.6.2 KHAOS main algorithm extension for coordinated motion . . 80
4.6.3 Coordinated handover results 87
4.7 Conclusion 95

Conclusion **97**

Bibliography **101**

Acronyms

B-Spline Non-Uniform Cubic B-Spline. iii, 7, 37, 38, 46, 47, 48, 49, 50, 51, 52, 55, 97, 98

FCW Flying CoWorker. iii, iv, 1, 2, 3, 4, 5, 6, 7, 9, 10, 15, 18, 20, 22, 23, 25, 26, 27, 28, 29, 32, 34, 35, 38, 39, 52, 53, 54, 55, 57, 58, 59, 60, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 112

HRI Human Robot Interaction. 1, 2, 9, 11, 47, 49, 62, 98, 112

KHAOS Kinematic Human Aware Optimization-based System for reactive planning of flying-coworker. iii, iv, 7, 9, 10, 13, 15, 17, 21, 22, 32, 34, 35, 37, 38, 39, 40, 41, 43, 45, 46, 47, 49, 50, 51, 52, 53, 54, 55, 57, 58, 80, 82, 85, 86, 91, 94, 95, 97, 98, 99, 112

Chapter 1

Introduction

Contents

1.1 Context 1
1.2 Flying CoWorker project 2
1.3 Publications 6
1.3.1 Published 6
1.3.2 Submitted 6
1.4 Manuscript organization 7

1.1 Context

Aerial drones use is increasing in our society, with new applications in human-populated areas which go beyond leisure and visual inspection. With increased payload and interaction capabilities, they are now considered for object delivery and collaboration with workers in civil and industrial applications. Safety is paramount in these environments, a trajectory error, a broken propeller, a poor estimation of obstacles, outdoor weather conditions or even low batteries, can have dramatic consequences on the health of humans in the surrounding area. In another perspective, navigation and interaction in close proximity to humans call for the consideration of some specific social or ergonomic skills, such as producing legible and acceptable motions.

Aerial drones or Unmanned Aerial Vehicles (UAVs) come in many shapes, weights and sizes depending on their intended use. Some can be equipped with a fixed or flapping wing, which are more suitable for long distance and high altitude flights. They can be found in the military, transport or scientific research fields. In this thesis, we are interested in another category of aerial drone whose lift is provided by several rotors. Control of the vehicle motion is achieved by varying the relative speed of each rotor to change the thrust and torque produced by each. Multi-rotor drones allow for hovering and high precision at a small size for use in human-populated environments.

Like ground mobile robots, multi-rotors drones can be used to navigate among humans. A recent survey by Mavrogiannis et al. [Mavrogiannis 2021] examines the work over the past three decades on social robot navigation. This survey shows a low amount of contributions related to drone navigation in the field of Human

Robot Interaction. However, due to the numerous additional constraints on aerial robots, the techniques applied to ground mobile robots are not directly applicable.

Multi-rotor drones can be equipped with a manipulator arm to perform tasks requiring physical contact. We speak then of autonomous aerial manipulator (AAM) combining the flexibility of 3D motion of multi rotor drones and object grasping using their arm. They can be used for industrial maintenance in difficult to access areas or to accomplish more specific tasks such as object manipulation or door opening.

In this thesis, we use an AAM to interact with humans in several aspects. First of all, to move in an environment populated with humans but also for a more specific purpose which is handover. In our context, a handover concerns the transfer of an object between a robot and a human. Handover task require to interact in proximity and physically with a human. Safety in these cases is essential but not sufficient to ensure good interaction. The human feeling of safety is also essential. A survey by Rubagotti et al. [Rubagotti 2022] talks about "Perceived Safety in Human Robot Interaction" (pHRI) and shows a growing number of contributions on drones in this field. On the other hand, for many years, contributions on mobile manipulators have remained low. Nowadays, the use of mobile manipulators is still mostly reserved for industrial or collaborative applications [Sandakalum 2022] where the robot does not enter the close space of the human.

All this shows how much remains to be explored in the field of autonomous aerial manipulators that navigate and physically interacting with humans as presented in this thesis. In this manuscript we present preliminary planning results in the interaction motion of an aerial manipulator named "Flying CoWorker". Further studies are needed to determine the levels of safety and ergonomics acceptable to interacting humans with AAMs. Also, we propose configurable solutions to be adapted according to the knowledge of Human Robot Interaction and in relation to the Flying CoWorker design.

1.2 Flying CoWorker project

The Flying CoWorker project is an ANR project[1] resulting from the collaboration between two French laboratories, the "Laboratoire d'analyse et d'architecture des systèmes du Centre National de la Recherche Scientifique" associated with the "Université de Toulouse" and the "Institut National de Recherche en Informatique et Automatique" (INRIA) associated with the "Université de Lorraine" and the CNRS.

LAAS-CNRS team The team "Robotique et InteractionS" (RIS) of the LAAS-CNRS realizes researchs in robotics and in particular for human/robot interaction,

[1]This work was partially supported by the French National Research Agency (ANR) (project Flying Co-Worker, `https://www.laas.fr/projects/flying_coworker`, grant ANR-18-CE33-0001) and the Artificial and Natural Intelligence Toulouse Institute - Institut 3iA (ANITI) under grant agreement No: ANR-19-PI3A-0004.

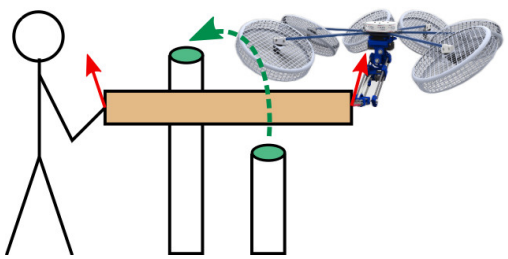
motion planning, aerial robots and control. To have available world-level robots and experimental means, the LAAS regrouped these resources in the robotic platform that allows to share the expertise and maintain up to date robots and systems. The interactions induced by robots are varied, and constitute the main interest of the Robotic and InteractionS (RIS) team involved in this project. The work of this thesis took place within the RIS team which is a leading group in HRI, aerial robotics and aerial manipulation systems. Antonio Franchi, a research associate at LAAS-RIS, is now professor at the University of Twente in the Netherlands. He is an expert in the control and design of aerial manipulators and part of the EU horizon 2020 project "AERIAL-CORE".

INRIA team The LARSEN team of LORIA is involved in this project and works on life-long autonomy and interaction skills for robots, combining robotics and AI / machine learning. The team has strong experience in activity recognition, multi-sensor fusion, planning under uncertainty, human tracking, robot learning and human-robot interaction. LORIA ("Lorraine Research Laboratory in Computer Science and its Applications") is a research unit (UMR 7503), common to CNRS, the University of Lorraine and INRIA. INRIA is a public scientific institute with 8 research centers in France.

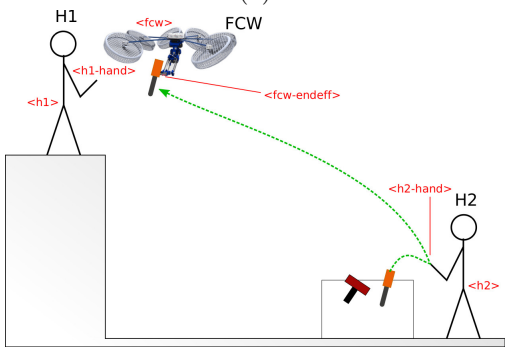
Context and scenarios Combining recent advances in the fields of human-robot physical and decisional interaction and control of aerial manipulators, the project investigates the Flying CoWorker, an aerial manipulator robot that cooperates with a worker to carry a bar or perform manipulation tasks. The ability of robots to consider humans and their safety is at the core of this research to build robots that can cooperatively manipulate and deliver objects to a worker in a safe, efficient, relevant and acceptable manner. The methods developed for terrestrial robots are not transposable to aerial manipulators because the base is unstable, the payload is limited and the energy constraints are strong. Based on the perception and interpretation of human activity, the objective of the project is to build an aerial manipulator capable of planning and controlling its movements to perform collaborative tasks.

One collaborative task envisaged is to carry a long object at one end from a point A to B in collaboration with a human carrying the other end as shown in Fig. 1.1a. Another collaborative task envisaged is the reception and transport of an object, such as a tool, in an environment populated by humans as illustrated in Fig. 1.1b with the aim of exchanging the object with a worker on arrival. There will therefore be a navigation phase with the tool as a load and a delicate phase at proximity of a worker with the aim of a handover (Fig. 1.1c).

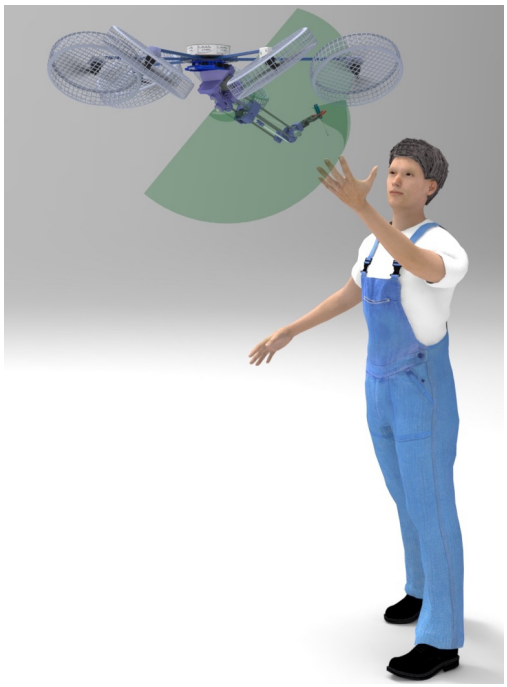
Hardware and low level control The Flying CoWorker available at LAAS is a unique prototype composed by a hexarotor with tilted propellers that is equipped with a 3-DoF arm as shown in Fig. 1.2. The whole mechanical design and low level control have been developed at LAAS, thus ensuring full mastering, indepen-



(a)



(b)



(c)

Figure 1.1: On the left: Flying CoWorker project scenarios of the bar transportation in collaboration with a human and navigation in human populated environment carrying a tool. On the right: Illustration of the Flying CoWorker platform in close proximity to a human for handover

dence, and high customizability. The tilting of the propellers confers to the robot the unique capability of exerting a total force in any direction in body frame (normal multi-rotors can exert their force in only one direction in body frame). The architecture thus allows for great stability of the base and very precise control, making it possible to act close to humans with greater safety. This unique feature confers 9 fully-actuated flying DoFs, thus overcoming the underactuation of standard aerial vehicles and making the FCW the state-of-the-art solution for mastering aerial physical interaction. LAAS already has gathered a long experience with such platform and its variants, one of which has been successfully shown at the Hannover fair 2017 as one of the finalist projects of the Kuka Innovation Award 2017.

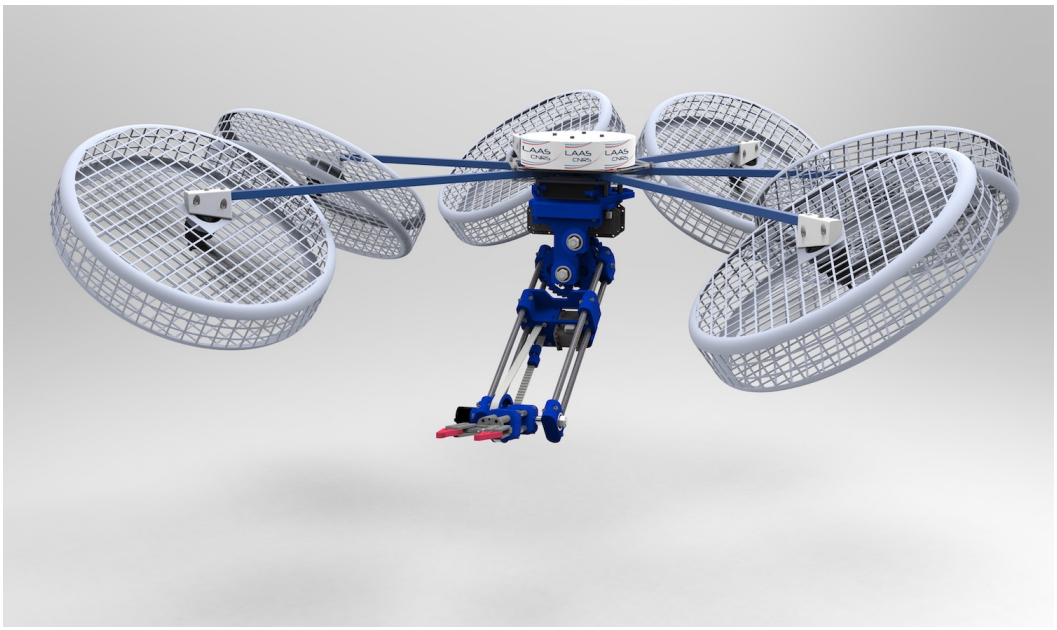


Figure 1.2: Flying CoWorker platform: hexarotor with tilted propellers equiped with a 3-DoF arm.

Project organization The tasks to be performed within the framework of the FlyingCoworker project require the development of technological bricks at all levels, ranging from perception to low-level control of the robot. To meet the many challenges of the project, the work is divided into four main workpackages corresponding to the work of four PhD students briefly presented below:

1. **WP1 - Physical human-robot interaction for a FCW** is focus on developping a general control framework for FCW physical interaction with human including force estimation for cooperative transportation of the bar and object handover.

2. **WP2 - Perception and interpretation of human activity** deals with

the human perception and its goals is to provide the information about the human that are relevant for the other WPs. The tasks explored are human perception by a FCW, activity recognition and prediction, gesture and intention estimation and finally predict the human activity.

3. **WP3 - Human-aware motion planning and control for a FCW** is the work presented in this manuscript. The challenge here is to develop a reactive planner that will allow the FCW to plan and execute adaptively the motions needed by the different collaborative tasks.

4. **WP4 - Task planning and collaborative task achievement** aims to design and build decisional processes that are necessary for a FCW to perform cooperative task achievement in a pertinent and fluent manner.

The activities of these four workpackages are linked. WP1 allows the low-level control of the Flying CoWorker actuators from the force sensors present on the arm and the recognition of human activity delivered by WP2. The planning of the Flying CoWorker motion is done by WP3 which provides the reactive trajectories to the controller developed in WP1. WP3 also uses the human activity data given by WP2 to take it into account during the planning. WP3 corresponds to the work presented in this thesis manuscript. Similarly this data is used by WP4 to coordinate the activity and plan the high level tasks of the Flying CoWorker.

1.3 Publications

Some of the contributions of this thesis have been published and listed hereafter:

1.3.1 Published

- **TRUC, Jérôme**, SINGAMANENI, Phani-Teja, SIDOBRE, Daniel, IVALDI, Serena et ALAMI, Rachid. KHAOS: a Kinematic Human Aware Optimization-based System for Reactive Planning of Flying-Coworker. In : ICRA 2022. 2022.

- **TRUC, Jérôme**, SIDOBRE, Daniel, et ALAMI, Rachid. KHAOS improvements using BSpline. In : ICCAS-The International Conference on Cognitive Aircraft Systems. 2022.

1.3.2 Submitted

- **TRUC, Jérôme**, SIDOBRE, Daniel, et ALAMI, Rachid. Reactive Planning for Coordinated Handover of an Autonomous Aerial Manipulator. Submitted to International Conference on Human-Robot Interaction (HRI). 2023.

1.4 Manuscript organization

Chapter 2 introduces KHAOS a human-aware reactive planner for social navigation of a Flying CoWorker. It considers human activity and the kinematic constraints of the Flying CoWorker to generate a trajectory in a reactive manner. KHAOS takes into account the dynamic environment in which the Flying CoWorker evolves to adapt in real time.

Chapter 3 presents the improvements made to the initial input path of KHAOS and the output trajectory it generates. We show the interest of the use of B-Spline allowing an efficient smoothing facilitating the respect of the Flying CoWorker kinematic constraints.

Chapter 4 presents an extension of KHAOS to provide a human-aware Flying CoWorker trajectory for the handover case. For this task, we present a way to determine the orientation of the Flying CoWorker base to signal to the human the intention to interact. We show how the final state of the Flying CoWorker is estimated for a handover. Finally, we present a way to plan the coordinated motion of the Flying CoWorker to achieve a smooth and acceptable motion for the human.

Chapter 5 concludes the work done during this thesis and describes the perspectives for improvement.

Chapter 2

Human-aware reactive navigation planning for a Flying CoWorker

Contents

2.1 Introduction . . . 9
2.2 Related work . . . 10
2.3 KHAOS . . . 12
2.3.1 Initial path . . . 13
2.3.2 Human-aware Costs . . . 15
2.3.3 Kinematic constraints . . . 17
2.3.4 KHAOS main algorithm . . . 17
2.3.5 Local and global trajectory costs . . . 18
2.3.6 Convergence criterion . . . 20
2.4 Results . . . 21
2.4.1 Simulation environment . . . 22
2.4.2 Examples . . . 23
2.5 Discussion and future work . . . 34
2.6 Conclusion . . . 35

The focus of this chapter is to show how we manage the navigation problem of the Flying CoWorker. We present KHAOS: a Kinematic Human Aware Optimization-based System for reactive planning of flying-coworker published and presented at ICRA 2022 conference [Truc 2022]. More details and discussions are given in this manuscript. Then, we describe our simulation environment especially how humans and obstacles are considered. This simulation environment will be also used in the following chapters. Finally, we present several situations to show that the behavior of our system can be used for Human Robot Interaction.

2.1 Introduction

In this chapter, we address navigation planning in the scenario of the Flying CoWorker project presented in 1.2: a multi-rotor drone that collaborates with workers to fetch small objects. The robot must fly in a human-populated area, where

only a handful of human workers may be "aware" of the robot's current task and mission, and a fraction of them may be involved in the physical interaction (e.g., object delivery): the robot must assume that most humans are "observers", i.e., they ignore its current mission and are not involved with it. In such conditions, the Flying CoWorker needs to carefully plan its 3D motion in a reactive way to navigate and act safely in close proximity to humans. Beyond safety, the Flying CoWorker should aim at exhibiting navigation strategies that are, as much as possible, socially aware: for example, it should avoid fast movements close or toward humans which could scare the observers; it must also maximise its visibility from the workers' point of view to avoid the surprise effect and improve the legibility of its intentions, particularly during an interaction.

We propose a Kinematic Human-Aware Optimization System (KHAOS) for reactive navigation planning which addresses the requirements mentioned above by synthesizing trajectories in the 3D space satisfying the kinematic constraints of the Flying CoWorker and ensuring the visibility and ease of the humans present in the environment. The human-aware behavior is realized by proposing a visibility cost and a novel discomfort cost which are combined with the kinematic constraints. A stochastic optimization process inspired by the STOMP algorithm [Kalakrishnan 2011] is used to optimise these various costs and thus generate the trajectory. These definitions of costs and constraints for the social navigation of the drones, together with the new reactive planning system, KHAOS, are the main contributions. In this work, we only consider multi-rotor drones such as the Flying CoWorker base, and throughout this chapter, the term drone always refers to a multi-rotor drone or Flying CoWorker.

2.2 Related work

Human-aware navigation A mobile robot that has to perform a defined task may have to navigate in its environment. It follows a trajectory starting from a position, for example its current position, and ending at the position of its goal. This trajectory is previously planned by a planner considering the environment as well as the obstacles in it before being executed by the robot. During its execution, the trajectory can be regularly updated or re-planned considering new events such as humans moving in the scene for example. This is called a reactive planner. The trajectory can be planned based on relative elements like the human-robot distance instead of considering an absolute position in the scene.

Human-aware robot navigation needs to consider additional constraints on the trajectory as well as the motion of the robot [Kruse 2013] to navigate safely around the humans. Most of the human-aware navigation planners mainly use only the so-called proxemics [Kruse 2013, Rios-Martinez 2015] criteria. This criterion often corresponds to a distance to be respected in order not to violate the virtual personal space around a human as proposed by Edward T. Hall [Hall 1966]. The work of Ferrer. et al. [Ferrer 2013], further developped by Repiso et. al. [Repiso 2017] uses

a social force model (SFM) based controller to navigate in the crowd and to accompany humans. Social force models represent moving agents like robots or humans as masses under virtual gravitational forces. Truong et. al [Truong 2017] extended the social force model to human-object and human-group interactions by proposing the proactive social motion model. Detecting interactions between people or between a person and an object, it allows the robot to avoid passing between people talking to each other for example. Inspired by these, Garell et. al [Garrell 2017] proposes an Aerial Social Force Model (ASFM), a 3D version of SFM, that allows the drones to safely accompany humans to their final destination. Some recent works in ground robot crowd navigation use Graph Convolutional Networks to predict human motion [Chen 2019]. Deep reinforcement learning techniques are used by Güldenring et al. [Guldenring 2020] to learn acceptable navigation behaviors and control a mobile robot in environments with active pedestrians. A recent contribution uses optimization to produce more legible robot trajectories along with modality shifting to address multi context navigation [Singamaneni 2021]. Based on elastic bands, the planner can, for example, switch between a single-band mode, for the robot only, and a dual-band mode, one for the robot and one for the human. In case of drone navigation, the recent work of Garell et. al [Garrell 2019] focuses on using neural networks to learn the non-linear ASFM to address the problem of human accompaniment. Unlike to these approaches based on reactive controller based on the Aerial Social Force Model, we present a reactive planning approach in this chapter inspired by the STOMP algorithm [Kalakrishnan 2011] which is highly flexible and can be adapted to various situations.

Interaction When the robot needs to approach a human for interaction, new motion criteria such as “approach the user from the front“ [Dautenhahn 2006, Koay 2007] and at a reduced speed, [Butler 2001] need to be introduced to take into account the specific constraints related to Human Robot Interaction. The noise and the wind generated by the propellers of drones cause significant additional annoyance for people as mentioned in [Cauchard 2015]. A user study carried out by Duncan et al. [Duncan 2013] evaluating the approach distance and height of the drone towards a human concluded that the human-human proxemics might not be directly transferable to human-aerial robot interactions. Yeh et al. [Yeh 2017] also performed a user study for evaluating proxemics in human-drone interaction and showed that the personal space of the humans varied based on social cues, like greeting and the design of the drone. A more recent work by Jensen et al. [Jensen 2018] studied the drone's interaction distance with a human to signal its presence and concluded that humans feel acknowledged between 2 m to 4 m. We use these distances to define the size of a visibility grid in the rest of this chapter.

At this point, one could also wonder what is the best angle of approach to interact with a human as studied for a mobile robot by Koay et al. [Koay 2007] who show a preference of the users for a frontal approach, in the visual field of the human. In our work, we want to propose safe and friendly trajectories for the

drone's navigation that are less disruptive to the humans it comes into contact with or encounters on its way. For this, we take into account the field of view of humans, the effort to make the drone visible and the discomfort caused to humans by the drone's motion.

The interaction between two agents, human or robot, can be improved when one knows the intention of the other. Pertinent communication of intentions by the robot can improve legibility [Dragan 2013] and safety. These intentions can be communicated by a more 'readable' trajectory of the robot as discussed by Dragan et al. [Dragan 2013] or with some gestures [May 2015] or through gaze [Khambhaita 2016, Hart 2021]. A recent work by Bevins and Duncan [Bevins 2021] studies the human perception of different drone paths. They show how the participants would react physically, as well as their perception of the messages contained in these flight paths. They generated several types of paths and, based on a three-phase user study, proposed some guidelines on path design to help robots communicate through their motion. Works by Kruse et. al [Kruse 2014] and Sisbot et al. [Sisbot 2007] studied the effect of directional costs and visibility to produce more legible paths for robot navigation. The approach presented by Khambhaita et al. [Khambhaita 2017] and later developed by Singamaneni et al. [Singamaneni 2020] proposed proactive trajectory planning for cooperative human-robot navigation and introduced the concept of time to collision (*time_to_collision*) that defines the time it would take for the robot to collide with an obstacle if it continues its movement at the same speed. It is used as a cost predicting a future collision with a human and pushing the robot to act earlier and show its intention to the human. The *discomfort_cost* proposed in this chapter is inspired by this. Similar behavior was applied to the case of the drone in [Yoon 2019], showing the anticipation effect where the drone takes into account the perception of the human.

In addition to the shape of the trajectory, the intention can also be deduced through the kinematics of the trajectory. A sudden motion towards the human can be equated with hostile intent, for example. The results of the study by Szafir et al. [Szafir 2014] show the importance of taking into account the phases of acceleration and deceleration of the drone and therefore, its kinematics to improve their social integration in an environment where they collaborate with humans.

2.3 KHAOS: a Kinematic Human Aware Optimization-based System for reactive planning of flying-coworker

We need a reactive planner which can perform well in a 3D environment that is not too sensitive to different local minima. We propose an approach inspired by the STOMP algorithm [Kalakrishnan 2011] which allows great flexibility due to its stochastic nature. The optimization takes as input a list of points forming an initial path whose ends are the start position and the desired goal. From this initial path,

it generates K noisy trajectories exploring the surrounding space and calculates the associated costs. These K trajectories are then analyzed using different costs applied at each timestep or waypoint to sort the most interesting configurations and finally generate an optimized trajectory.

In the context of this chapter, a trajectory is a list of waypoints with a time stamp. Each waypoint has a three-dimensional position and may have associated kinematic information. The transformation to a time-continuous trajectory is discussed in the next chapter

In this section, we first propose a way to generate an initial path to provide as input of the system. We then introduce the various costs we use to help the optimizer generate safe and comfortable trajectories for humans in the scene. We also present how we extend the original STOMP algorithm to take into account not only the kinematic constraints of the robot but also the constraints imposed to respect human comfort and safety. Finally, we discuss how we evaluate the trajectory cost at different stages of the optimization process.

2.3.1 Initial path

KHAOS, like STOMP from which it is inspired, needs an initial path as a starting point which is then deformed by the optimization process using the various applied costs. The initial path is composed of a list of waypoints whose number is configurable.

Chronologically, the initial path is planned before the drone starts its motion. KHAOS generates a first trajectory from the initial path and then the drone starts to execute this trajectory. Then the trajectory being executed serves as a new input to KHAOS to replan and update the trajectory considering new events like dynamic obstacles or human motions.

The definition of the initial path is therefore a preliminary step to the reactive planning phase carried out by KHAOS. It is a step that serves as a guide for the reactive planner at start-up, so it is essential to respect certain criteria. The initial path must above all avoid obstacles and take into account the geometry of the drone to respect a safe distance. Furthermore, the length of the path between the starting point and the target must be minimal to favour short duration trajectories.

Before starting its motion, humans can accept that the robot takes a few moments to plan a first trajectory. We can therefore afford to use an initial path planner which takes time to calculate the initial path but which will explore the environment extensively to find the most interesting solution.

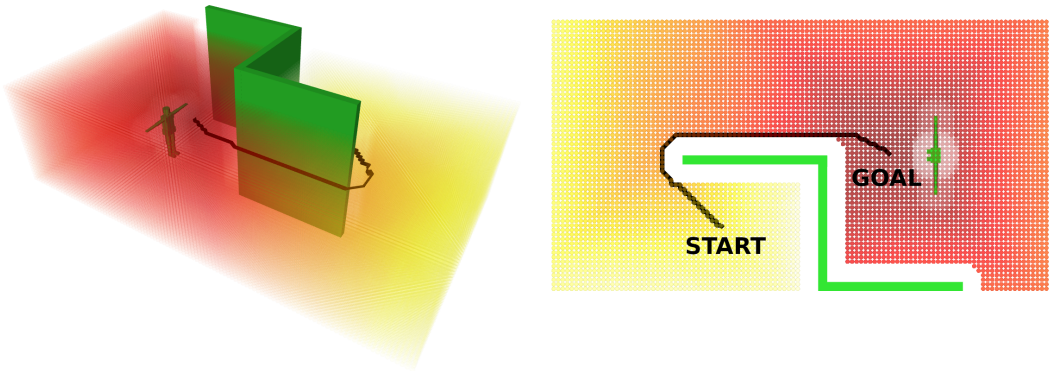
For this purpose, we propose below a planner that analyses the collisions in a large space of the environment by a 3D grid and finds the shortest path taking into account the obstacles.

Implemented method To meet the input requirements of KHAOS, we use a wavefront expansion algorithm applied to a three dimensional grid. To do this, a cost is propagated through the grid starting from the cell of the grid corresponding

to the position of the target the robot is to reach. Each cell is a cube of configurable size. For a given grid cell, the cost is for example incremented by a value of 1 for the six closest cells. For the cells in the corners of the cube surrounding the original cell, the cost is incremented by $\sqrt{3}$ and by $\sqrt{2}$ for the remaining cells. If the tested cell is located too close or in an obstacle, then the cost applied is -1 corresponding to a value that will be ignored in the path finding phase. Going through the grid by applying these costs makes it possible to obtain significant costs for the most distant regions by considering the obstacles.

The initial path is then obtained by starting from the cost associated with the cell located at the starting point of the robot and performing a gradient descent to reach the objective which corresponds by default to the lowest cost of the grid.

The list of cells recorded during the gradient descent is used to obtain the list of positions forming the initial path. These positions correspond to the midpoint of the cells. An example of the obtained result can be seen in Fig. 2.1 where the starting point is in a region where the robot is hidden by walls and the objective is in front of the human represented in green. The darker colors correspond to the lower costs and the lighter colors correspond to the higher costs. The lowest costs are therefore found around the goal. The highest costs correspond well to the most distant regions given the obstacles because where the starting point is, it is necessary to go around the wall to access this area.



(a) 3D view (b) top view

Figure 2.1: Initial path generated using wave expansion algorithm. Darker colors correspond to the lower costs and the lighter colors correspond to the higher costs.

Discussion The method used is simple and interesting to obtain a result that avoids local minima, is repeatable and considers obstacles in the scene. On the other hand, it has several shortcomings.

First of all, due to the nature of the grid, the result obtained is a discretized path, it is thus necessary to apply a smoothing method to have a continuous path. We will see in the next chapter how we overcome this problem. Second, a global

planner is usually slow, so it takes some time to update and compute a new path if objects or humans in the scene move. This is why it is necessary to use other techniques like the one used in KHAOS. Finally, the initial path generated using this technique only optimizes the distance to travel. To improve the system, it would be interesting to integrate human-aware costs such as those integrated in KHAOS. This could improve the shape of the initial path, which could further improve the performance of KHAOS in its trajectory search.

2.3.2 Human-aware Costs

Below we define the various social and kinematic constraints that are used in the optimisation phase of KHAOS.

We first present a means of evaluating the discomfort caused to humans through the *discomfort_cost*. Then we present the *visibility_cost* which reflects how the human's visual field is taken into account but also the effort required to turn the head and see the Flying CoWorker. Finally, we present a way to constrain the speed of the Flying CoWorker according to a *discomfort_constraint* while considering the kinematic limits of the Flying CoWorker.

2.3.2.1 Discomfort

To represent the discomfort caused to one or more humans when a drone moves in the environment, we consider a cost based on the relative speed and distance between the drone and humans present. Indeed, we want to translate the fact that a drone moving fast and close to a human is much less comfortable than a distant slow drone. In addition, if the drone is forced in one way or another to pass close to a human, it must adapt its speed by reducing it to limit the discomfort generated to this human, or even stop if it has reached a certain threshold. The *time_to_collision* cost presented in [Khambhaita 2017] is the first part of the answer, but we need also information on this cost even when the velocity vector of the drone is not oriented towards the human. Therefore, we formulate the *discomfort_cost* as:

$$C_{dis} = \frac{\|\overrightarrow{V_{rob}} - \overrightarrow{V_{hum}}\|}{Dist_{rob-hum}} + \frac{\alpha_{proximity}}{Dist^2_{rob-hum}} \tag{2.1}$$

where $\|\overrightarrow{V_{rob}} - \overrightarrow{V_{hum}}\|$ and $Dist_{rob-hum}$ are respectively the relative speed and distance between the drone and the human. The second term of Eq. (2.1) describes the cost associated with the proximity of the drone to the human whose influence can be adjusted using the scaling factor $\alpha_{proximity}$. It reflects the nuisance caused by the presence of the drone hovering in the human environment. This can include the physical presence of the drone but also the noise emitted by the propellers.

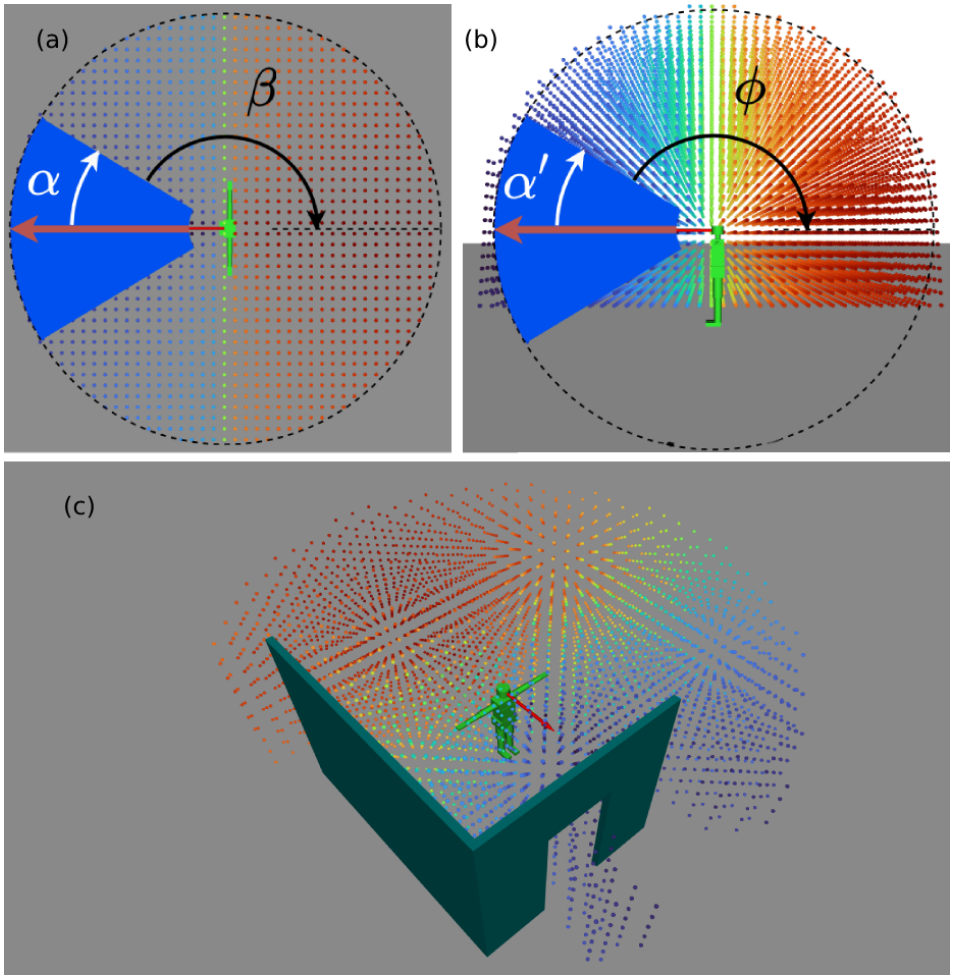


Figure 2.2: Visibility cost : a) (resp. b) Cutaway top (resp. side) view showing the 3D simulation rendering of the change in visibility cost as a function of the panoramic (resp. tilt) angle. c) 3D view showing the consideration of obstacles. The human gaze direction is represented by a red arrow. Red colored cells correspond to high cost while blue colors correspond to low cost.

2.3.2.2 Visibility and effort to see

The visibility criterion presented in [Sisbot 2007] uses a 2D grid. Since a drone can move in three dimensions, we need to extend the model to a 3D grid. A 3D grid centered at the origin of the human visual field gives the visibility cost, C_{vis}. For each cell the cost is computed as follows:

- All cells contained in a cone of pan and tilt angles respectively equal to $2 * |\alpha|$ and $2 * |\alpha'|$ (blue zone in Fig. 2.2a and Fig. 2.2b) located in front of the human have the same cost value. This value is relatively low ($= 1$) representing a relatively free zone and corresponding to the preferential approach zone according to [Koay 2007].

- Angles increase clockwise as $|\beta| = \pi - |\alpha|$ and $|\phi| = \pi - |\alpha|$. Black and white arrows in Fig. 2.2 start at the minimum value of the angle and end at the maximum values. Cells in the zone starting at $|\alpha_{max}|$ (respectively $|\alpha'_{max}|$) and ending at $|\beta_{max}|$ (respectively $|\phi_{max}|$) have a cost proportional to $|\alpha|+|\beta|$ (respectively $|\alpha'|+|\phi|$). This makes it possible to reconcile the approach zones in the visual field which are less comfortable than the frontal zone and the zones which are not in the visual field of the human that require an effort to turn around.

- Zones hidden by obstacles, out of the visual field and beyond a threshold distance (4 m) correspond to a zero cost value.

As depicted in Fig. 2.2, the orientation of the human gaze is represented by the red arrow. Directly in front of human is represented in dark blue, the zone of low cost, and then there is a gradual increase of the cost more and more towards the rear corresponding to more and more reddish colors.

2.3.3 Kinematic constraints

The driving idea behind our approach is to take into account the kinematics of the drone while respecting human comfort by taking as a reference the *discomfort_cost* defined above. The particularity of the STOMP algorithm on which we based the optimization part of our approach is the generation and comparison of K noisy trajectories. These noisy trajectories aim to explore space around the previous trajectory in a stochastic manner.

Starting from this principle, we propose in Algorithm 1 to constrain these K generated trajectories by considering the kinematic constraints of the drone such as its maximum speed v_{max}, acceleration a_{max} and deceleration dec_{max}. To this, we add an additional constraint linked to the cost of discomfort by setting a value DCF_{max}. This value cannot be exceeded and is named as *discomfort_constraint* in this manuscript. Thus for each point of the trajectory generated randomly, we calculate the maximum speed v_{kin} that the drone can reach considering its kinematics limits. If for this position and speed, the C_{dis} exceeds *discomfort_constraint* limit DCF_{max}, then the *discomfort_constraint* predominates and limits the speed below the maximum speed that it is possible to achieve. Conversely, if for a given position, the drone can move at its maximum speed and it is not inconvenient for humans, then it will limit its speed to its maximum kinematic limits.

2.3.4 KHAOS main algorithm

The initial path and costs defined above are used by the KHAOS main algorithm which is based on the well-known STOMP algorithm [Kalakrishnan 2011]. Algorithm 2 represents the different steps that take place when running KHAOS. Like STOMP, it starts from a starting state x_0 and an ending state x_N, which in the context of this chapter represent the Cartesian coordinates $(x_{Base}, y_{Base}, z_{Base})$ of

Algorithm 1: Constrained velocity computation

Given

- Drone kinematics constraints: v_{max}, a_{max}, dec_{max}
- *discomfort_constraint*: DCF_{max}

for *each noisy smooth trajectory* **do**
  for *each 3D position of the drone* **do**
    Compute kinematic velocity of the drone v_{kin};
    Compute discomfort cost C_{dis} with v_{kin};
    if $C_{dis} < DCF_{max}$ **then**
      $v_{choice} = v_{kin}$;
    else
      $v_{choice} = (DCF_{max} - \frac{\alpha_{proximity}}{Dist^2_{hum-rob}}) * Dist_{hum-rob}$;
    end
    $v_{drone} = min(v_{max}, v_{choice})$
  end
end

the centre of mass of the Flying CoWorker base. The initial path obtained by the method described in Section 2.3.1 allows to define the initial discretized trajectory $\boldsymbol{\theta}$ composed of N Cartesian positions. As presented by Kalakrishnan et al. [Kalakrishnan 2011], $\boldsymbol{N}$ is the number of waypoints chosen to discretize the trajectory, $\boldsymbol{A}$ is a finite differencing matrix that when multiplied by the position vector $\boldsymbol{\theta}$, produces accelerations $\ddot{\boldsymbol{\theta}}$. $\boldsymbol{M}$ is used in the optimization process to smooth the updated trajectory.

As the initial path only gives the positions for the base, Algorithm 1 is first applied to define the speeds associated with $\boldsymbol{\theta}$ before optimisation. These values are needed to evaluate the cost of the path to be compared during the convergence test detailed later in this chapter.

The first step in the optimisation loop is to create K noisy trajectories $\tilde{\boldsymbol{\theta}}_i$ using a normal distribution $\mathcal{N}$ with mean $\boldsymbol{\theta}$ and variance $\boldsymbol{R}^{-1}$. For each noisy trajectory, speeds are calculated using Algorithm 1 as the next step is to estimate the costs for each waypoint on each noisy trajectory. The speeds are needed to evaluate the local costs and especially the *discomfort_cost*. The next steps allow to sort the costs using $P(\tilde{\boldsymbol{\theta}}_{k,i})$ and to keep the most interesting base positions before estimating the global cost of the trajectory $Q(\boldsymbol{\theta})$.

2.3.5 Local and global trajectory costs

Similarly to the STOMP algorithm [Kalakrishnan 2011], we estimate the costs in two distinct stages of the algorithm. A first step where an estimated local cost for a particular waypoint is estimated. A second step carried out at the end of the

Algorithm 2: KHAOS main algorithm

- **Given:**
  - Start (x_0) and goal (x_N) states, $x_i \in \mathbb{R}^{DoF}$
  - An initial discretized trajectory $\boldsymbol{\theta} \in \mathbb{R}^{DoF \times N}$
  - A state-dependant cost function $q(x_i)$
- **Precompute:**
  - $\boldsymbol{A}$ = finite difference matrix
  - $\boldsymbol{R}^{-1} = (\boldsymbol{A}^T \boldsymbol{A})^{-1}$
  - $\boldsymbol{M} = \boldsymbol{R}^{-1}$, with each column scaled such that the maximum element is $(1/N)$
  - Apply Algorithm 1 to $\boldsymbol{\theta}$ to obtains associated speeds, $\boldsymbol{\theta} \in \mathbb{R}^{2 \times DoF \times N}$
- **Repeat until convergence of global trajectory cost $Q(\theta)$:**
  1. Create K noisy trajectories, $\tilde{\boldsymbol{\theta}}_1 \cdots \tilde{\boldsymbol{\theta}}_K$ with parameters $\boldsymbol{\theta} + \boldsymbol{\varepsilon}_k$, where $\boldsymbol{\varepsilon}_k = \mathcal{N}(0, \boldsymbol{R}^{-1})$
  2. Apply Algorithm 1 to each $\tilde{\boldsymbol{\theta}}_i$ to obtains associated speeds
  3. For $k = 1 \cdots K$, compute:
     (a) Local cost $S(\tilde{\boldsymbol{\theta}}_{k,i}) = q(\tilde{\boldsymbol{\theta}}_{k,i})$ according to Equation 2.2
     (b) Probability metric for sorting: $P(\tilde{\boldsymbol{\theta}}_{k,i}) = \frac{e^{-\frac{1}{\lambda} S(\tilde{\boldsymbol{\theta}}_{k,i})}}{\sum_{l=1}^{K} [e^{-\frac{1}{\lambda} S(\tilde{\boldsymbol{\theta}}_{l,i})}]}$
  4. For $i \cdots (N-1)$, compute: $[\delta\tilde{\boldsymbol{\theta}}]_i = \sum_{k=1}^{K} P(\tilde{\boldsymbol{\theta}}_{k,i})[\boldsymbol{\varepsilon}_k]_i$
  5. Compute $\delta\boldsymbol{\theta} = \boldsymbol{M}\delta\tilde{\boldsymbol{\theta}}$
  6. Update $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \delta\boldsymbol{\theta}$
  7. Compute global trajectory cost $Q(\boldsymbol{\theta}) = \sum_{i=1}^{N} q(\boldsymbol{\theta}_i) + \frac{1}{2}\boldsymbol{\theta}^T \boldsymbol{R} \boldsymbol{\theta}$ according to Equation 2.3

optimisation loop to estimate the global cost of the trajectory.

Local cost The local cost function estimated at each waypoint is defined as follows:

$$S(\tilde{\boldsymbol{\theta}}_{k,i}) = \begin{cases} \sum (C_{vis} + C_{dis} + C_{time_local}) & \text{if no collision} \\ C_{obstacle} & \text{else} \end{cases}$$
$$C_{time_local} = \alpha_{time_local} * \frac{dist_to_goal}{||V_i||} \tag{2.2}$$

where C_{vis} is the visibility cost defined in 2.3.2.2, C_{dis} is the *discomfort_cost* and $C_{obstacles}$ is the obstacle cost, which pushes the waypoints away from the obstacles whenever possible without violating the social constraints.

Note the introduced cost C_{time_local} which is homogeneous to a time. The calculated distance *dist_to_goal* corresponds to the distance between the considered waypoint and the objective, passing through all the following waypoints of the trajectory. $||V_i||$ is the drone speed magnitude for the considered waypoint and α_{time_local} is a constant. This cost therefore represents the time it would take the Flying CoWorker to reach the objective if it continued at the same speed. Reducing this cost favours high speeds or short distances.

Global cost In contrast to the local cost, we estimate here a cost for the whole trajectory called global cost. It is used to estimate the whole trajectory and compare it to the input trajectory or to the trajectories that did not meet the convergence criterion.

The global trajectory cost is defined as follows:

$$Q(\boldsymbol{\theta}) = \sum_{i=1}^{N} (C_{vis} + C_{time}) + C_{smooth}$$
$$C_{time} = \alpha_{time} * \frac{L}{V} \tag{2.3}$$

where, N is the number of waypoints and C_{vis} is the visibility cost defined in 2.3.2.2. C_{time} is a cost that can be assimilated to the duration of the trajectory to be executed by the drone when cumulated over the whole trajectory. L and V are the distance and average speed of the drone between waypoint i and $i-1$ respectively and α_{time} is a constant. $C_{smooth} = \frac{1}{2}\boldsymbol{\theta}^T \boldsymbol{R} \boldsymbol{\theta}$ represents the sum of squared accelerations along the trajectory as defined in [Kalakrishnan 2011] and allows to improve the smoothness of the trajectory.

2.3.6 Convergence criterion

The optimization loop detailed above is executed as long as the convergence criterion is not met. Many convergence criteria are listed in the literature and several of them have been tested for our case. Nevertheless, it is always difficult to define a convergence criterion that is a good compromise between execution speed and

quality of the result, i.e. the trajectory in our case. Our need is to obtain first of all a trajectory which does not include any waypoint in collision with the obstacles and the humans present in the environment. Secondly, we want the trajectory to be sufficiently deformed if necessary, to respect the various social costs, to bypass the obstacles widely enough to avoid the surprise effect for example. Finally, we want the execution time to be short enough to take advantage of a reactive planner that can react quickly to various changes in the environment and to unforeseen events.

Algorithm 3: Convergence criterion

Given

- Counter initialization: $attempt \leftarrow 0$
- Maximum number of attempts: $max_attempts$
- Previous global trajectory cost: $Q(\boldsymbol{\theta})_{previous}$

while $attempt < max_attempts$ **do**
  $attempt++$;
  Compute new global trajectory cost: $Q(\boldsymbol{\theta})$;
  if $Q(\boldsymbol{\theta}) < Q(\boldsymbol{\theta})_{previous}$ **then**
    Reset counter: $attempt \leftarrow 0$;
    Update global cost: $Q(\boldsymbol{\theta})_{previous} \leftarrow Q(\boldsymbol{\theta})$;
  end
end

To meet our need, we propose to use Algorithm 3 where a counter called *attempt* is used to list the number of successive attempts where the optimization process did not improve the global trajectory cost $Q(\boldsymbol{\theta})$ previously defined. If the optimizer succeeds in improving the global cost of the trajectory, then the *attempt* counter is reset to 0. If, on the other hand, after $max_attempts$ attempts without success, the optimizer is stopped to keep the trajectory corresponding to the best attempt. Thus, by changing the $max_attempts$ parameter, the desired quality of the trajectory can be set.

Another interesting alternative may be to give a time criterion to the optimizer as used by Park. et al. [Park 2012]. By setting a maximum time for the optimization loop, it is possible to be sure that the result is always obtained within the given time. This can be interesting when integrating into a state machine for example. It is thus possible to control the duration of the optimization task and avoid blocking the system if for some reason the optimization loop takes much longer than expected.

2.4 Results

In this section, we first present the simulation environment used to challenge KHAOS. Then, several situations are studied, each of these have their own speci-

ficities to give a global view of the capacities of KHAOS. Through these examples, we will also see the influence of the various costs defined in the previous section.

2.4.1 Simulation environment

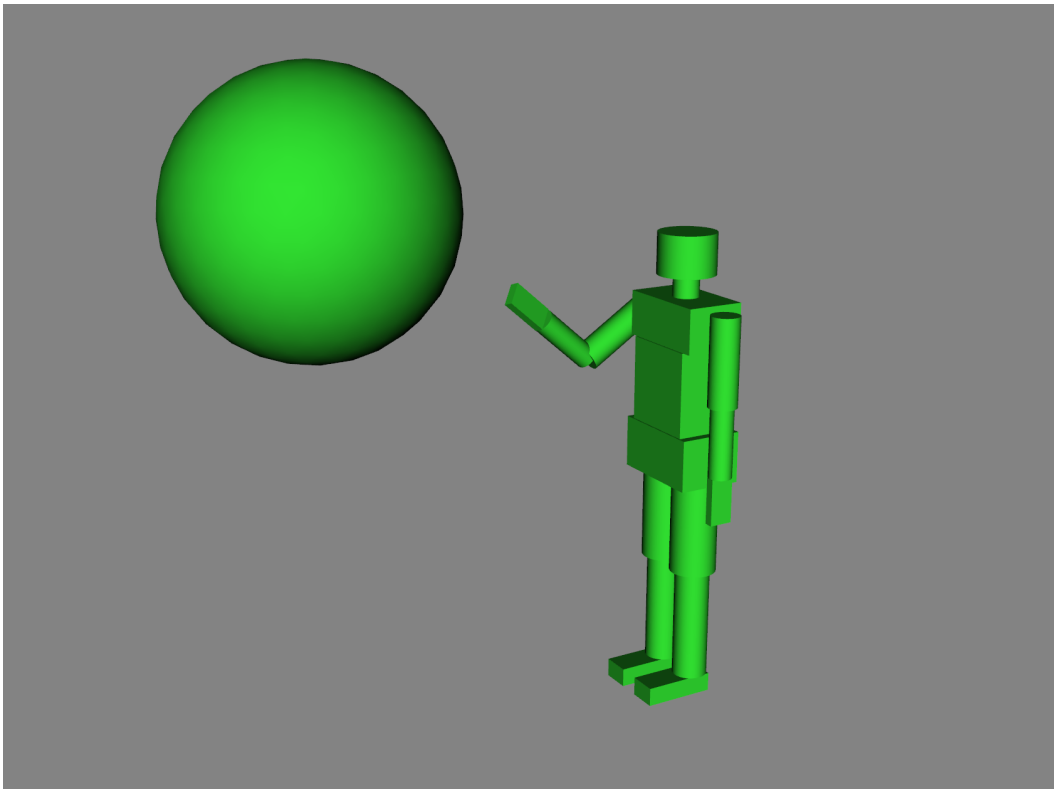


Figure 2.3: On the left: 0.9 m sphere diameter used by KHAOS to do collision tests for the Flying CoWorker platform. On the right: Human model composed of primitive convex objects.

To demonstrate the KHAOS behaviour, we need a suitable simulation environment. Our system is human-aware and therefore requires a human model to be tested as well as a way to simulate the robot itself. Many collision tests are carried out by KHAOS. To maximise the performance of collision tests, we use simple primitives rather than complex meshes to represent the human(s), the robot and the environment as collision objects.

The human model used is represented by a set of simple primitives such as cylinders and boxes as represented in Fig. 2.3. These primitive objects are considered as collision objects. Using this model, it is possible to differentiate between the different parts of the human body. For example, the position of the right hand can be known independently of the position of the head because they are two separate collision objects.

In this same figure, the collision object used for the collision tests of the Flying

CoWorker base can be seen. It is considered as a 0.9 m diameter sphere that can be configurable. This representation gives an idea of the imposing dimensions of the Flying CoWorker base in the proximity of the human. To improve readability, the sphere corresponding to the drone is not shown in the following figures in this section, showing only its trajectories. Depending on the examples, the scene is completed with obstacles and humans.

The simulation results presented in the rest of this chapter use the MoveIt [Chitta 2012] collision scene where the different collision objects of the human, the robot and the obstacles are imported.

2.4.2 Examples

2.4.2.1 Frontal approach

For this first example, we explore the case where the drone approaches the human very closely from the front as shown in Fig. 2.4. We represent the trajectory by drawing each of its segments with an arrow whose color depends on the drone average speed. The more the color tends towards red, the higher the speed and the more the color tends towards green, the slower the speed. From this representation, we can visually see the phases of acceleration and deceleration of the drone.

The drone starts at a distance of 9 m from the human and approaches him at a very close distance of 0.5 m and at a height of 1.5 m. This interaction distance is very small and close to the human's head, which can be considered a very uncomfortable situation for him. As the approach is frontal, the cost of the human visual field does not influence and does not distort the trajectory. In our implementation, we favor positions far from obstacles, and that explains the slight deformation of the trajectory towards the opposite direction of the ground, which is considered as an obstacle.

***discomfort_constraint* of 0.5** We first choose a *discomfort_constraint* equal to 0.5 that can be imagined as the maximum speed of $1\,\mathrm{m\,s^{-1}}$ for the drone at a distance of 2 m from the human. Fig. 2.4 represents the speed of the drone as well as the *discomfort_cost* along the trajectory. The movement of the drone can be broken down into three phases. First of all, the drone accelerates in accordance with its kinematic limits until it reaches its maximum speed of $1\,\mathrm{m\,s^{-1}}$. Gradually approaching the human, the *discomfort_cost* increases until it reaches the *discomfort_constraint* of 0.5 that we have set. Once this maximum value is reached, the optimizer will regulate the speed and start to decelerate so as not to violate either the *discomfort_constraint* or the kinematic constraints of the drone.

***discomfort_constraint* of 0.25** Let us now consider the same trajectory but this time we fixed the *discomfort_constraint* at 0.25. In Fig. 2.4f, we find the same acceleration phase as before because the drone is far enough away. On the other hand, the *discomfort_constraint* is reached more quickly which pushes the drone to

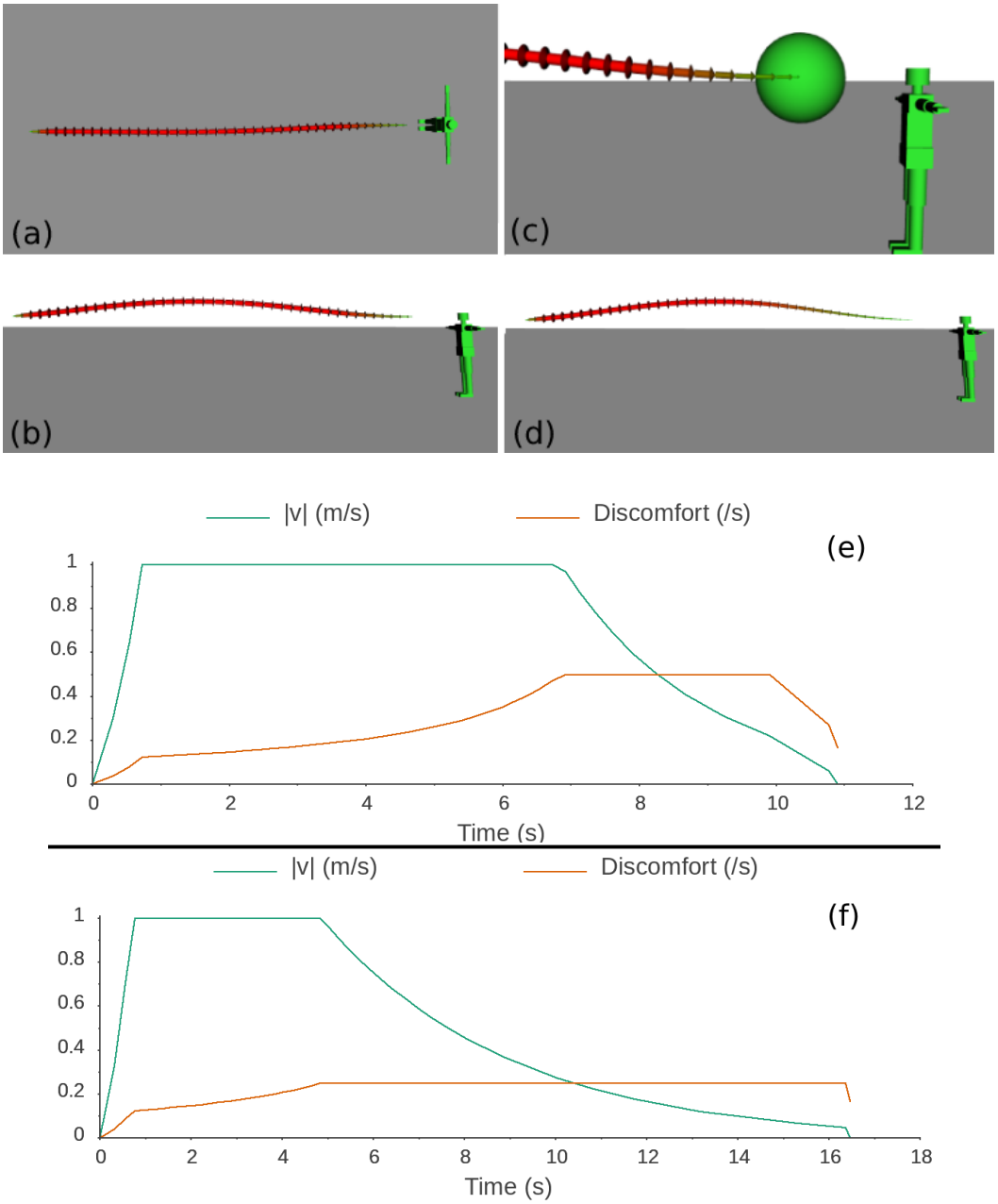


Figure 2.4: **Frontal approach with a *discomfort_constraint* of 0.5**: a) Top view b) Side view c) Side view with a zoom at the top showing the drone's size represented by a sphere of 0.9 m e) Drone speed and *discomfort_cost* over time. **Frontal approach with a *discomfort_constraint* of 0.25**: d) Side view f) Drone speed and *discomfort_cost* over time.

slow down approximately 2 s earlier, allowing the drone to signal to the human its intention to slow down in his/her presence. In addition, the deceleration phase is

much longer by around 3 s, which gives time to the human to better adapt to the presence of the drone.

The *discomfort_constraint* is a parameter that allows the robot's behaviour to be adapted to the situation. The operator can accept a faster approach, for example if the robot has already come several times under the same conditions. Or on the other hand, if we want a very smooth deceleration, we can simply reduce this value.

2.4.2.2 The FCW overtake the human

Now let's take a look at how the optimizer behaves when the human visual field has a strong influence. We propose to visualize this influence Fig. 2.5 through an example where the Flying CoWorker starts at a distance of about 6 m from the human, approaches him in a straight line while passing to his/her right at a close distance of 1.5 m and stops at a distance of about 6 m behind his/her back. As in the previous use case, the height at which the Flying CoWorker moves in its original and non-optimized trajectory is 1.5 m.

Fig. 2.5b and Fig. 2.5c show the shape of the optimized trajectory for a *discomfort_constraint* of $0.5\,s^{-1}$, while Fig. 2.5d shows the corresponding Flying CoWorker speed magnitude and the *discomfort_cost* along the trajectory. The red line represents the original path provided to the optimizer.

Here, the shape of the trajectory is greatly distorted by the *discomfort_cost* and the human visual field. Having no obstacles, the optimizer explored the space and found a trajectory where it could maximize the Flying CoWorker's speed while reducing the *visibility_cost* to a minimum. We also notice the deformation of the trajectory along the z-axis, and the optimizer has indeed generated a trajectory moving away as far as possible from the human.

We can observe that the *discomfort_constraint* limit is never reached and the *visibility_cost* is, in this case, the major factor distorting the trajectory. Once the trajectory reaches a minimum *visibility_cost* and *discomfort_cost*, the only parameters which will influence the total cost of the trajectory are the length and speed which will limit the deformation of the trajectory and prevent the Flying CoWorker from going too far away.

Changing the *discomfort_constraint* to a value of 0.25 shows that the deformation can be greater as illustrated in Fig. 2.5a, where the shape of the trajectories is visually compared to Fig. 2.5b. In this case, the influence of the visibility cost is identical, but the *discomfort_cost* pushes back the trajectory more strongly in order to minimize the C_{time_local} and C_{time} defined in 2.3.5. This ensures that the speed is maximised along the trajectory and therefore the travel time is minimised while respecting human comfort.

2.4.2.3 Corridor

Until now, the trajectories generated by the optimizer are not subject to constraints linked to the environment except the human himself. Now let's study a similar

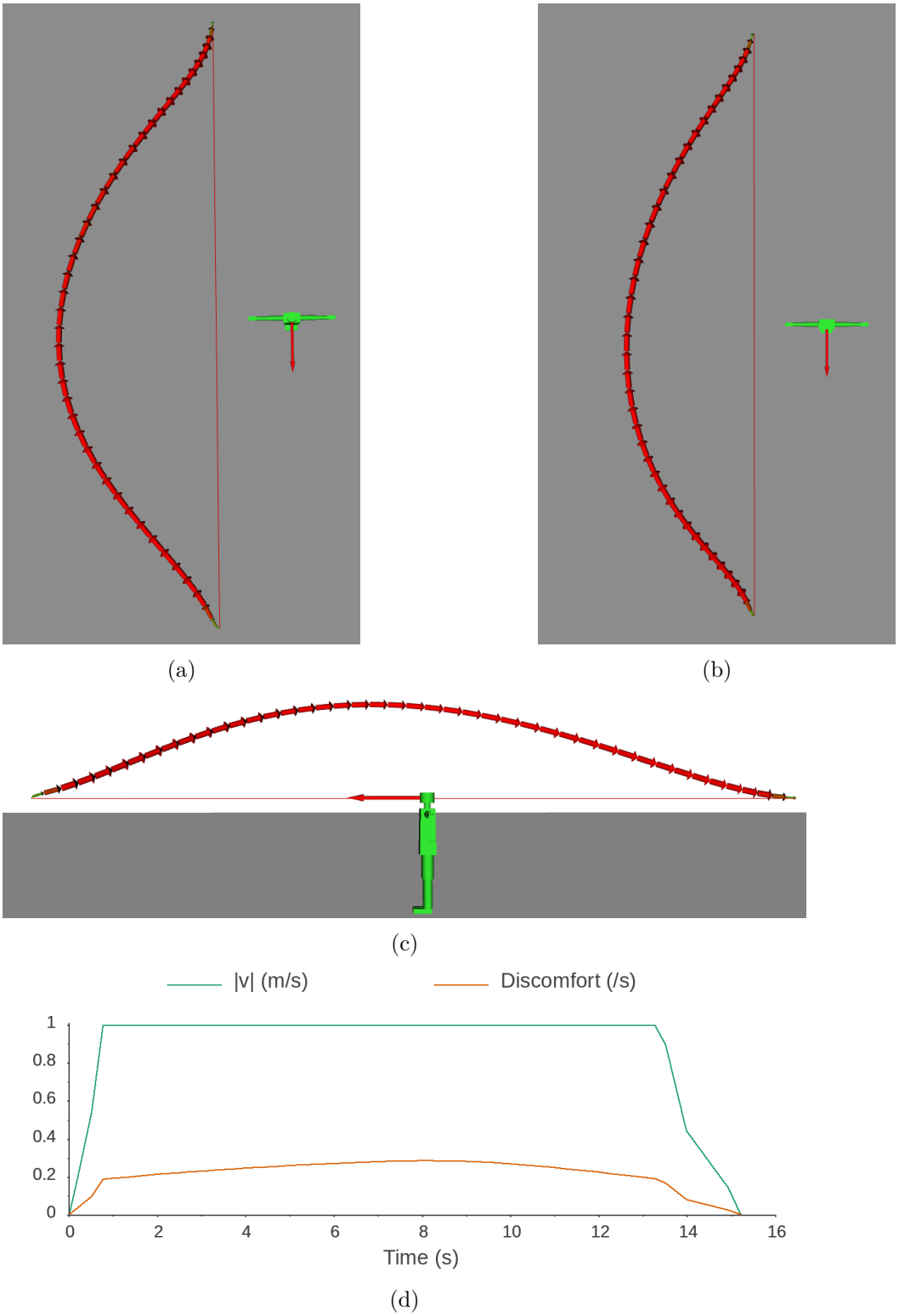
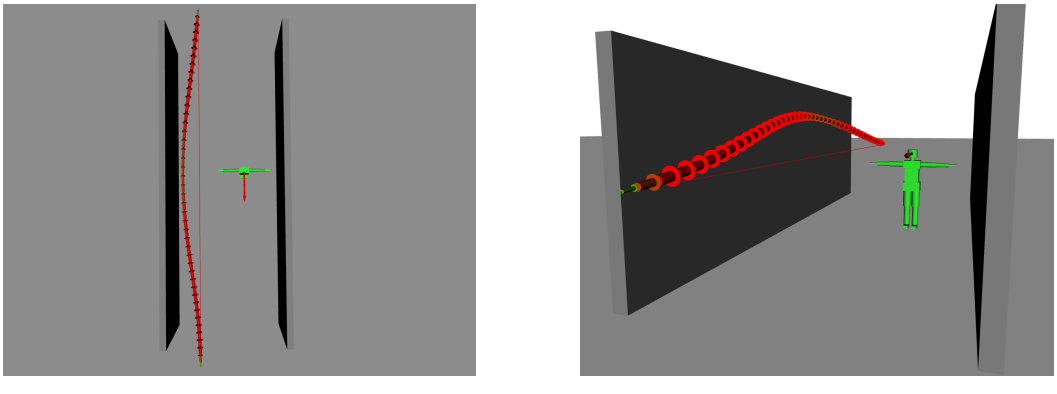
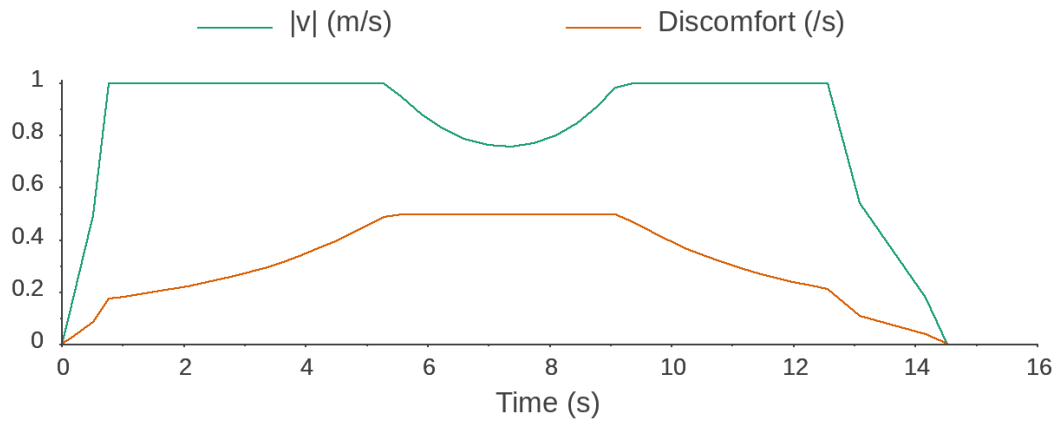


Figure 2.5: Flying CoWorker overtaking the human with a *discomfort_constraint* of $0.25\,\mathrm{s}^{-1}$ (a) and $0.5\,\mathrm{s}^{-1}$ (b) with the according side view (c). d) Flying CoWorker speed magnitude and *discomfort_cost* as a function of time for a *discomfort_constraint* of $0.5\,\mathrm{s}^{-1}$



(a) Top view (b) Side view



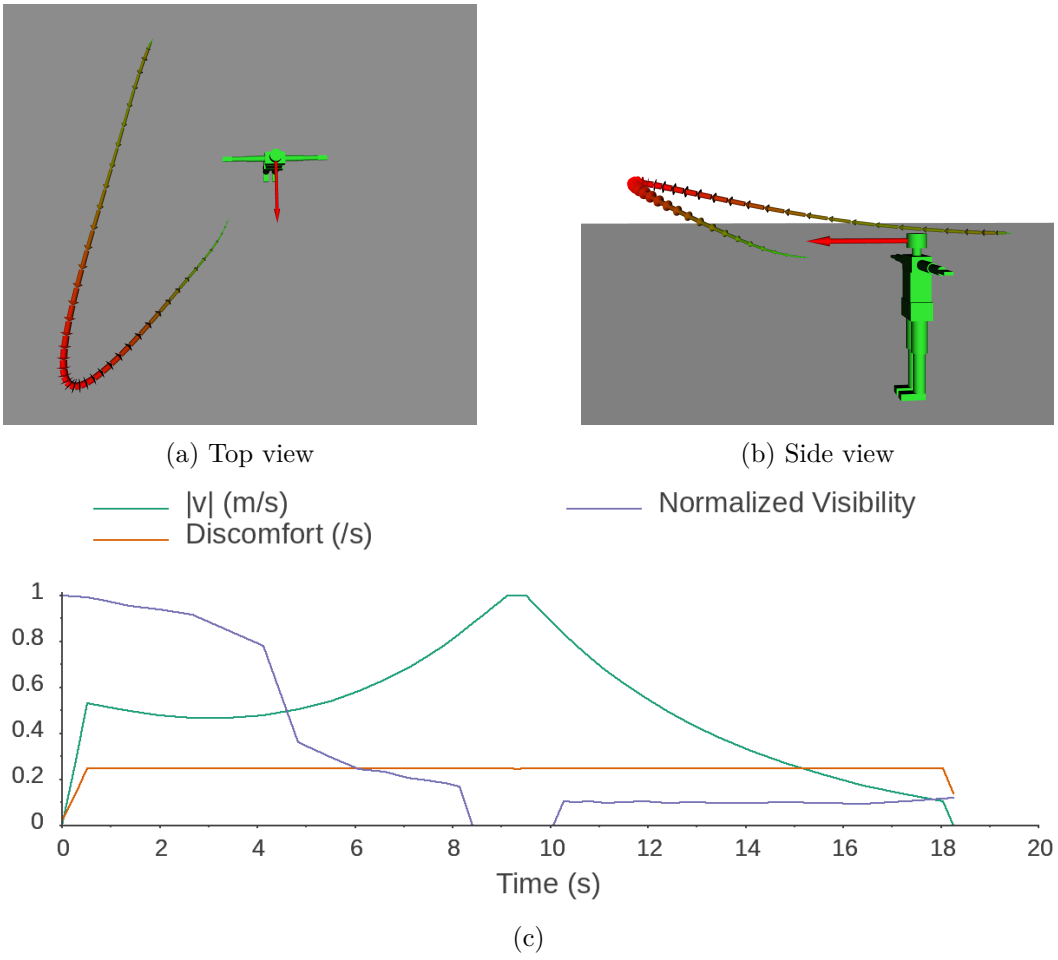
(c)

Figure 2.6: Flying CoWorker trajectory in the corridor in presence of the human for a *discomfort_constraint* of $0.5\,\mathrm{s}^{-1}$: a) Top view b) Side view c) Flying CoWorker speed magnitude and *discomfort_cost* as a function of time

situation where the Flying CoWorker navigates in a corridor and crosses a human on its way. For this, we added 2 walls 3 meters high, positioned so that there is sufficient space for the Flying CoWorker to navigate to the human's right. The walls are high enough to force the Flying CoWorker through the corridor instead of going around it. In this situation, the corridor walls do not allow the optimizer to generate trajectories that deviate greatly from the human in order to allow the Flying CoWorker to navigate at maximum speed. Here we use the same original path as in the previous cases in order to have better visual comparisons of the deformation of the trajectories.

The result is visible in Fig. 2.6 where we observe a trajectory with a *discomfort_constraint* of $0.5\,\mathrm{s}^{-1}$ that deviates much less from the human than in Fig. 2.5. The side view Fig. 2.6b also shows an upward deformation in order to increase the distance between the human and the Flying CoWorker.

Despite the constraints induced by the walls, the speed is close to the maximum speed along the trajectory except for the points closest to the human as we can see in



(a) Top view (b) Side view

(c)

Figure 2.7: Flying CoWorker trajectory starting behind the human's back for a *discomfort_constraint* of $0.25\,\mathrm{s}^{-1}$: a) Top view b) Side view c) Flying CoWorker speed magnitude, normalized *visibility_cost* and *discomfort_cost* as a function of time

Fig. 2.6c. For a *discomfort_constraint* of $0.5\,\mathrm{s}^{-1}$, the distance is generally sufficient along the trajectory and the speed does not need to be strongly corrected. Indeed, in Fig. 2.6c, we see that the *discomfort_cost* does not exceed $0.4\,\mathrm{s}^{-1}$, which means that from the point of view of the *discomfort_cost*, the points of the trajectory could be closer to the human while respecting the *discomfort_constraint* of $0.5\,\mathrm{s}^{-1}$. On the other hand, the *visibility_cost* pushes the points of the trajectory further away from the human. Here, the points of the trajectory are spatially blocked by the wall, and therefore the *discomfort_cost* takes over by limiting the speed if necessary. This is what we observe in Fig. 2.6c between 5 and 9 seconds when the speed is limited by the *discomfort_constraint*.

2.4.2.4 The FCW arrives from the back

Here we consider the case where the Flying CoWorker arrives from the back of the human. Its starting point is at the back of the human and close enough to be in the *visibility_cost* grid defined in 2.3.2.2. The *visibility_cost* at this position is therefore non-zero. This in order to show that our planner makes sure to limit the effort necessary for the human to turn his/her head and thus have the Flying CoWorker in his/her field of view while respecting the *discomfort_constraint*.

Fig. 2.7 shows that the Flying CoWorker will move away from the human to maximize its speed, reduce the effort required to see the Flying CoWorker while adapting its speed because it starts very close to the human.

In Fig. 2.7c we can see the time evolution of the normalized *visibility_cost* in addition to the speed and *discomfort_cost*. It can be seen that very soon the *discomfort_constraint* is reached, limiting the speed and forcing the Flying CoWorker to move away to gain speed. On the other hand, the *visibility_cost* keeps reducing until it reaches a zero value because the Flying CoWorker has moved so far away that it has gone out of the *visibility_cost* grid before entering it again. Then the *visibility_cost* remains relatively low because the approach is made in a favourable visibility cone for the human.

2.4.2.5 Two Humans

We now study the case where the robot meets a second human on its path. Fig. 2.8 shows the Flying CoWorker first navigating a corridor and passing a human represented in blue on its way, then continues by approaching a second human in green from the back to finish at a position where it can exchange an object with him.

To build such a corridor, we added two 3 m high blocks that materialize these walls, positioned so that there is sufficient space for the drone to navigate to the human's right in the corridor. We also add a ceiling to the corridor to force the drone through it instead of going around it. In order to make the scene visible in the figure, the block representing the ceiling has been made almost transparent.

Each human has his/her own *visibility_cost* grid computed as defined in 2.3.2.2. In the presence of several humans, for a given Flying CoWorker position, we consider only the highest *discomfort_cost* and *visibility_cost* values among these different humans for the optimization process.

In this situation, the corridor walls do not allow the optimizer to generate trajectories that deviate greatly from the human in blue located in the middle of the corridor. Despite the constraints induced by the walls, the speed is close to the maximum speed along the trajectory except for the points closest to the blue human. Here, the points of the trajectory are spatially blocked by the wall, and therefore the *discomfort_cost* takes over by limiting the speed. This is what we observe in Fig. 2.8c between 4 and 13 s when the speed is limited by the *discomfort_constraint*. At the same time, we can observe that the *visibility_cost* greatly increases because the drone goes more and more towards the back of the blue human

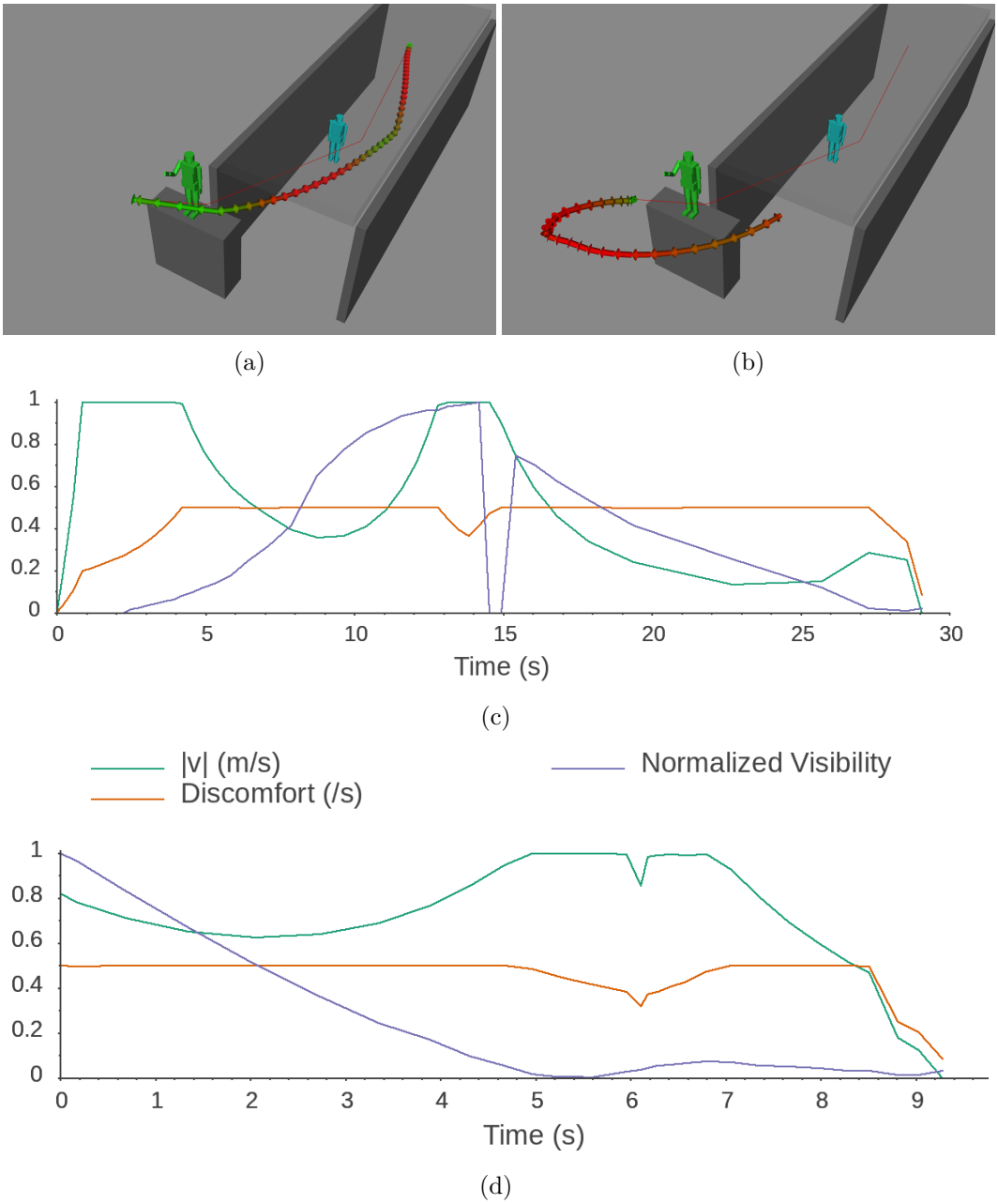
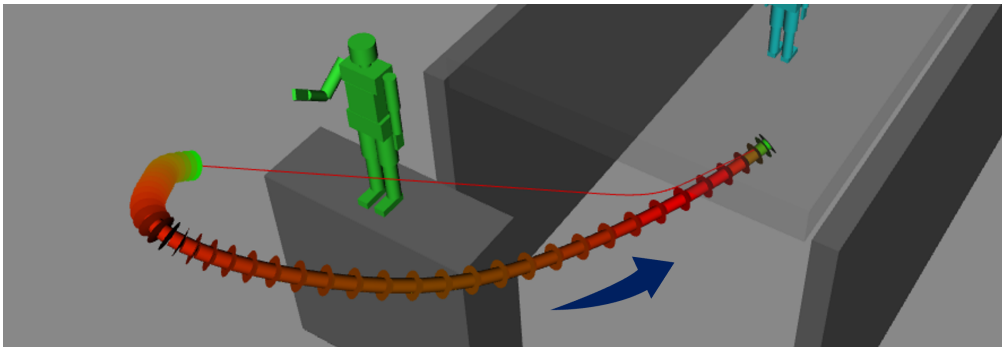


Figure 2.8: Trajectory execution in a two-humans scenario for a *discomfort_constraint* of 0.5. a) First iteration, the robot starts by navigating in the corridor b) A few iterations later, after having passed the blue human in the corridor c) (resp. d) speed magnitude, *discomfort_cost* and *visibility_cost* as a function of time corresponding to the trajectory from (a) (resp b: Time origin corresponds to the beginning of the re-planned trajectory).

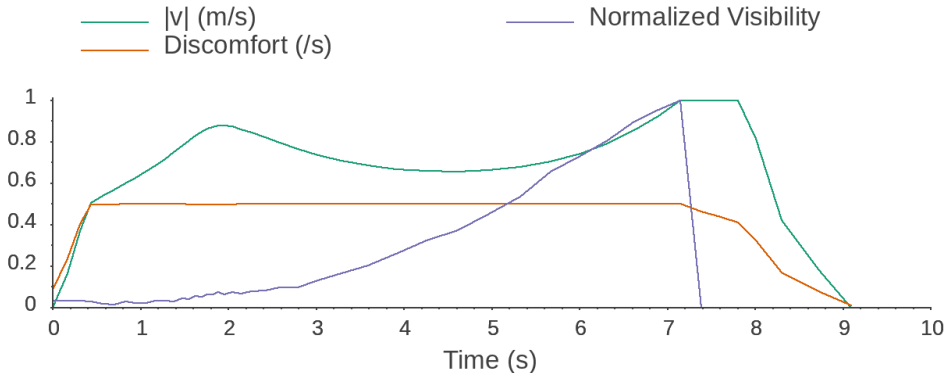
and can't pass far from him until it comes out of the influence of the *visibility_cost* grid at approximately 14.5 s and *visibility_cost* drop to 0.

Once the drone has passed the human in blue, it finds itself behind the back of another human represented in green. Here, the influence of his/her *visibility_cost* becomes predominant compared to that of the human in blue. The trajectory greatly deviates as shown in Fig. 2.8b and this shows how our planner ensures to limit the effort necessary for the human to turn his/her head, and thus have the drone in his/her field of view while respecting the *discomfort_constraint*. Fig. 2.8d shows that the drone moves away from the human by adapting its speed and reduces the effort required to see it specifically at the end when it is in close proximity to the human.

In the situation where the drone disengages from the human after handover (Fig. 2.9a), the shape of the trajectory is similar to the approach from the back. It first prioritizes the positions in front of the human to move away and go around the human. It accelerates smoothly, as shown in Fig. 2.9b until the first 2 s, limited by the *discomfort_cost*, and then adapts its speed in agreement with other constraints.



(a)



(b)

Figure 2.9: Drone disengagement from the human and motion toward the corridor with a *discomfort_constraint* of 0.5. a) Shape of the trajectory b) speed magnitude, *discomfort_constraint* and *visibility_cost* as a function of time. Blue arrow indicates the direction of the trajectory.

2.4.2.6 Planner reactivity

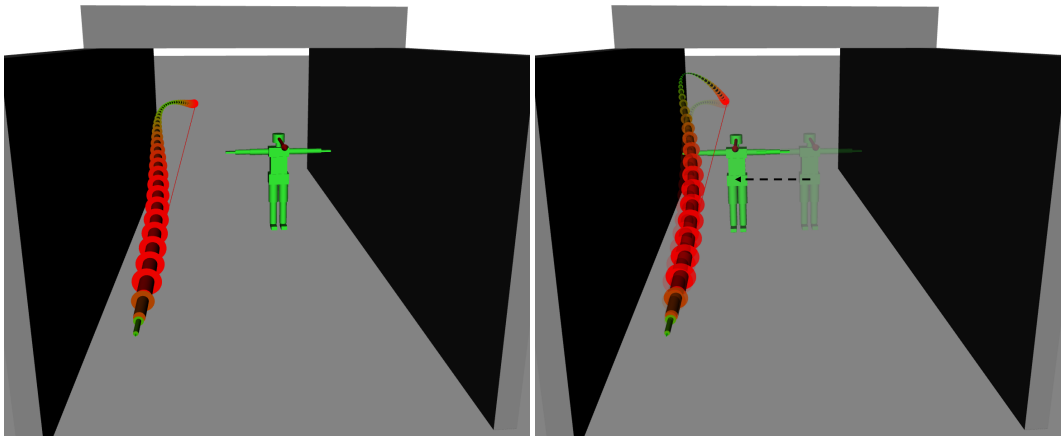
In order to illustrate the reactivity of our planner, we use the corridor scenario as used previously and study the reactivity of the planner when the human in the corridor moves. Fig. 2.10a shows the trajectory deformation with the *discomfort_constraint* fixed at 0.25 when the human is located on the right in the corridor at the beginning. While KHAOS is being executed, the human moves towards the opposite wall reducing the Flying CoWorker initial passage space. This can be seen in Fig. 2.10b where we observe that the planner reacts immediately by deviating the trajectory upward to avoid the human. This behaviour shows the ability of KHAOS to operate in real time.

The ceiling block above the corridor prevents the planner from deviating the trajectory further upwards. In return, the speed is immediately adjusted to respect the social constraints as can be seen in Fig. 2.10d especially when the Flying CoWorker is close to the human.

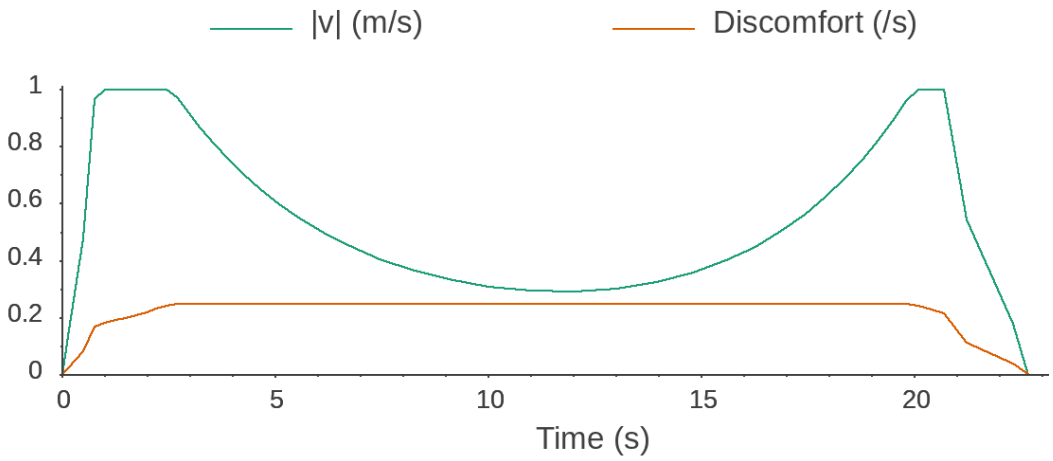
2.4.2.7 Highly constrained environment

Sometimes the environment does not allow for a satisfactory solution close to the original path that does not currently consider social constraints or simply because the environment has changed suddenly. Here we show that KHAOS has also the ability to adapt and find solutions even in very constrained environments.

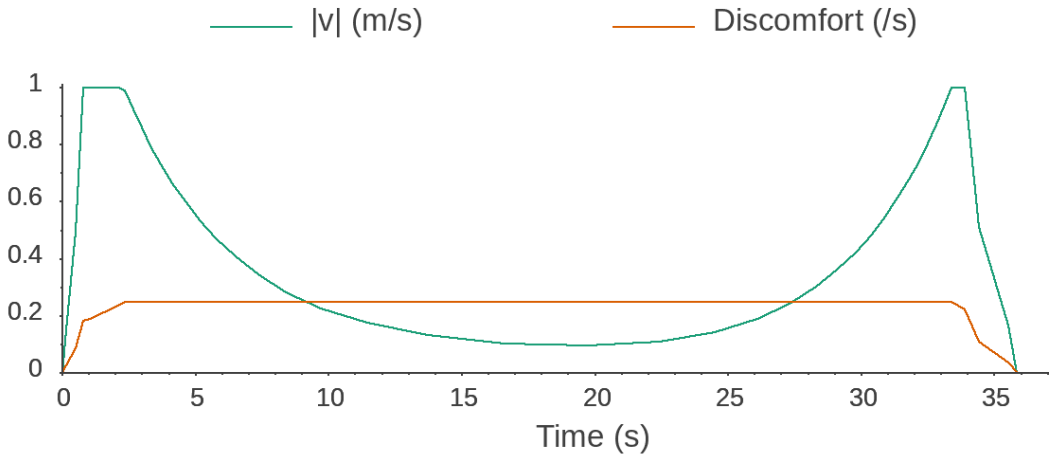
For that, we take the previous corridor scene and add an obstacle that can be compared to a counter in order to restrict the possibilities of the passage of the Flying CoWorker through the place where the human is located. It is placed on the ground and prevents access to the Flying CoWorker from the area below the human's right arm. The space between the ceiling and the human's head is sufficient for the drone to pass, but in this case, the trajectory would be very uncomfortable for the human or even dangerous. We deliberately challenge KHAOS by choosing an initial path passing through the area below the human's left arm represented by the red line in Fig. 2.11. Flying in this area would push it to pass very close to the human body, which would be very uncomfortable. If the Flying CoWorker had no other choice, the optimizer would be able to generate a trajectory where the drone would go at a very slow speed as it passes close to the human body. In the case presented here, the optimizer chooses a trajectory far from the human and on the opposite side by changing the homotopy class. We, therefore, can say that the trajectories generated by the optimizer are not deformed just locally but by exploring the surrounding space to find more suitable solutions. This trajectory is not only more comfortable for the human but also allows the drone to reach its goal more quickly.



(a) (b)



(c)



(d)

Figure 2.10: Planner reactivity when the human is moving to his/her right for a *discomfort_constraint* of 0.25. On the left: First iteration (a) with corresponding drone's speed magnitude and *discomfort_cost* as a function of time (c). On the right: Next iteration just after moving the human on his/her right reducing the passage to the drone (b) with its corresponding speed magnitude and *discomfort_cost* as a function of time (d).

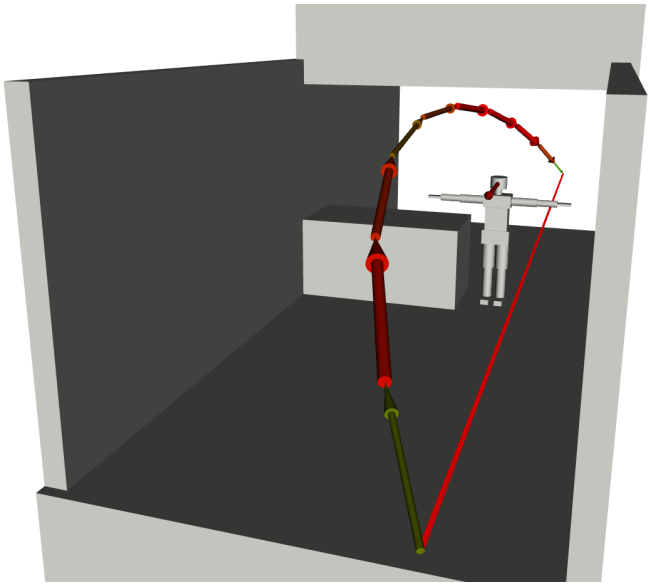


Figure 2.11: Drone's trajectory along the constrained corridor with a *discomfort_constraint* of 0.5.

2.5 Discussion and future work

The trajectory generated by KHAOS is refreshed at a frequency between 5-10Hz on a standard computer (1.9 GHz Intel i7 CPU, 32 GB of memory): with such a performance, our reactive planner can be used in real-time in real robot experiments. Trajectories used in our experiments are generated from an initial path corresponding to a straight line to simplify reading except for the two humans scenario. Indeed, in our case, we must be attentive not only to the shape of the trajectories but also to the adaptation of the speed.

The noise generated by a multi-rotor drone like the Flying CoWorker is a function of its speed and/or acceleration. Moreover, the acoustic intensity decreases proportionally to the inverse of the square of the distance. We can therefore be assumed that the *discomfort_constraint* described in this manuscript also reduces this nuisance.

The physical model used for the calculation of the speeds of the drone by considering the *discomfort_constraint* can be improved. At some parts of the trajectory the speed is not derivable, but most of the time, the system produces a smooth motion compatible with human interaction. To improve it, we can combine it with the bounded jerk model techniques [Sidobre 2019] that could produce smoother and better velocity profiles. This improvement will be presented in the next chapter.

We plan to improve the planner by adding the management of drone orientation. On the human-aware level, we want to go more in-depth by studying how to adapt the trajectories by integrating the handover phase [Mainprice 2010] as well as the control of the manipulator's arm [Sisbot 2012], which requires being very close to the human. We aim at a coordinated arm movement which can, for example, start

reaching to exchange an object while the navigation phase is not yet over, which requires testing deformable configuration spaces for the drone. The work on these topics is presented in Chapter 4.

2.6 Conclusion

We have presented in this chapter a planning system called KHAOS for the generation of reactive human-aware trajectories in 3D for a Flying CoWorker.

Our system takes into account the kinematic constraints of the multi-rotor drone. It also considers the potential discomfort caused by the drone's fast motion in proximity to humans. For that, we proposed a *discomfort_cost* considering the relative distances and speeds between a drone and a human. In addition, the visibility of the drone by humans and the human effort to see it are also taken into account. Finally, we have shown how the drone adapts its behavior in several situations. Through these examples, we have highlighted the importance and influence of the various social constraints defined at the beginning of the chapter. We have shown that our system respects the kinematic constraints of the robot in all situations in addition to respecting the comfort of the human.

Chapter 3

KHAOS improvements Using BSpline

Contents

3.1 Introduction 37
3.2 Issues 38
3.3 Tests of existing solutions 40
3.4 KHAOS improvements using B-Spline 46
3.4.1 Introduction to B-Spline 47
3.4.2 SoftMotion extension using Non-Uniform Cubic B-Spline . . 49
3.5 Results 51
3.5.1 Initial path smoothing 51
3.5.2 Kinematic compliance 52
3.6 Conclusion 55

3.1 Introduction

In the previous chapter, we presented KHAOS [Truc 2022], a Kinematic Human Aware Optimization-based System for Reactive Planning of Flying-Coworker in the context of the "Flying Co-Worker" project. It offers a solution that combines motion planning with respect for the robot's kinematic constraints. From an initial path composed of waypoints corresponding only to positions, our planner is able to modify this path by assigning new positions. These new positions are assigned a speed, all of which allows the human-aware and kinematic constraints of the robot used to be respected.

We first want to make sure that the kinematic information given at the output of KHAOS is feasible. KHAOS provides a list of waypoints, so we want to link them to obtain a continuous time trajectory executable by a low-level controller. We also want this trajectory to respect as much as possible the kinematic information given by KHAOS to avoid violating the social constraints between two consecutive waypoints.

In this chapter, we start by introducing the problematic related to the poor smoothing of the output trajectories presented in the previous chapter. We continue by testing two trajectory generators from the literature. These tests highlight the

need to develop a solution dedicated to the problem of defining a long trajectory for a complex system. An important characteristic of KHAOS is that the final trajectory is build by deforming an initial path and that the quality of this initial path has a large influence on the solution. The use of Non-Uniform Cubic B-Spline provides a solution to the two problems of the initial trajectory and the generation of a continuous trajectory. We propose the use of an extension of the softMotion library based on Non-Uniform Cubic B-Spline. Finally, we show through examples the improvement brought to the conversion of a KHAOS trajectory into a continuous trajectory in time.

3.2 Issues

Initial path quality As mentioned in the previous chapter, our planning system is inspired by the widely used STOMP algorithm. The STOMP algorithm optimises an initial path from a set of costs to generate a new smoothed path. This smoothing is dependent on the initial path, as STOMP does not smooth out any significant breaks in the initial path. These breaks will remain in the final result and may even be amplified.

By inheritance, KHAOS suffers from the same problem and therefore requires smoothing of the initial input path in order to obtain a satisfactory result. Furthermore, it generates a 3D trajectory constrained by kinematic limits (bounded speed, acceleration and jerk) and human-aware costs that may also have an impact on the kinematic parameters. This trajectory must therefore respect the feasibility from a kinematic point of view by avoiding, for example, too tight turns that would force the robot to slow down during execution.

Fig. 3.1 clearly shows the impact of the initial path defects on the output of KHAOS. The initial path shown in red has two very sharp turns. After several iterations, KHAOS was able to smooth out the turn visible on the left of the figure. The second one, visible on the right, is reflected in the planner output. The defects are still present and without improvement it is clear that the execution of such a trajectory will be inefficient from a kinematic point of view. The Flying CoWorker must reduce its speed sharply at each of these turns or even stop to follow the trajectory faithfully.

Compliance of shape and kinematic In order to link the waypoints generated by KHAOS and thus generate a complete continuous trajectory in time, we need a trajectory generator adapted to the constraints of our system for the following reasons:

- KHAOS output contains a large number of waypoints. A too small number of waypoints does not allow to obtain a sufficiently accurate result, especially when there are many turns or when the length of the trajectory is important. Because of the large number of waypoints, the distance between them can be quite small.

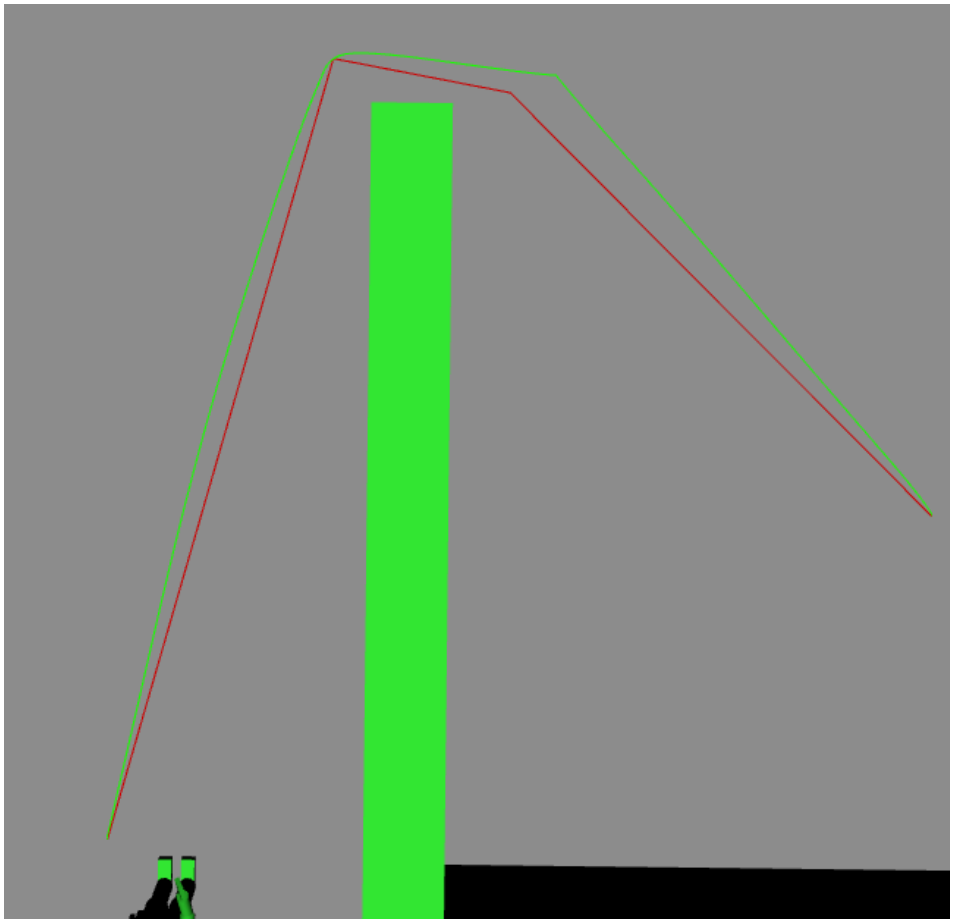


Figure 3.1: Shape of the initial path (red) and trajectory generated by KHAOS (green) after several iterations when no specific smoothing is applied

- The final trajectory must pass through the waypoints and connect them without violating the social constraints imposed by KHAOS. The social constraint most impacted by the trajectory generation is the *discomfort_constraint* defined in 2.3.2.1. Indeed, a too large trajectory deviation between two waypoints can cause an increase of the *discomfort_cost*. This increase can be due to an undesired physical approach of the Flying CoWorker to the human or an increase of the speed too important. It is therefore preferable to avoid that the speed of the Flying CoWorker between two waypoints is higher than the highest speed at these two waypoints.

- Each waypoint generated by KHAOS contains position and speed information. Speed magnitude is limited by the discomfort constraint and defined from a basic constant acceleration calculation model. The components of the speed vector are then defined from the segments connecting the previous and next waypoint. The most important information to be used by the trajectory generator is therefore the magnitude of the speed. Then, the components of

the speed vector must be adapted to ensure continuity.

3.3 Tests of existing solutions

Meeting the criteria defined in the previous section with existing solutions is not trivial. In our case, it is not enough to find a curve connecting two points with an acceleration phase, a cruising speed phase and a deceleration phase. The speeds for most of the different waypoints are not zero. In this section, we show in parallel the results of two interesting libraries intended to solve our problem:

- KDTP (for KinoDynamic Trajectory Planner), based on a steering method presented in the work of A. Boeuf [Boeuf 2017]. Spline-based, it is a trajectory generator with minimal time and constant instantaneity. It is applicable to any robot with decoupled (or differentially flat) dynamics to determine the proximity of the dynamic states of a quadrotor.

- Ruckig, the result of work by Berschei et al. [Berscheid 2021]. An algorithm for online trajectory generation (OTG) respecting third-order constraints and complete kinematic target states. Given any initial state of a system with multiple degrees of freedom (DoFs), Ruckig calculates a time-optimal trajectory to an arbitrary target state defined by its position, velocity, and acceleration limited by velocity, acceleration, and jerk constraints.

These two libraries thus make it possible to connect two states from kinematic data. We used them to connect the waypoints generated by KHAOS for two different trajectories. The first one is a straight line trajectory with an acceleration phase, a constant speed phase and a deceleration phase. The second is slightly deflected by an obstacle to force the trajectory to have a certain curvature.

The kinematic limits used to constrain the trajectory generators are those used by KHAOS. For speed, we limit to $1\,\mathrm{m\,s^{-1}}$, for acceleration we impose $0.5\,\mathrm{m\,s^{-2}}$ and $10\,\mathrm{m\,s^{-3}}$ for jerk. For the specific case of KDTP which requires a limit in snap, we have limited it to $30\,\mathrm{m\,s^{-4}}$.

Straigth line trajectory Fig. 3.2 shows the results for the straight line trajectory for both trajectory generators. On the left are the results from KDTP and on the right those from Ruckig. For each (Fig. 3.2a and 3.2b), we show a shot of the shape of the generated trajectories represented by red lines. The waypoints generated by KHAOS are represented by green spheres.

Overall, we can see that both trajectory generators try to pass through the different waypoints with a slight advantage for Ruckig. However, when zooming in on the start of the trajectories (Fig. 3.2c and 3.2d), we notice large unexplained trajectory deviations around the waypoints. These deviations are less important for Ruckig but still exist and make the trajectory impossible to execute. The speed magnitude curves (Fig. 3.2e for KDTP and Fig. 3.2f for Ruckig) show that the

trajectory generators have difficulties in adapting the speed between each waypoint. In both cases we observe oscillations between a zero speed and the speed provided by KHAOS (green curve). For KDTP, we notice that the speed is higher than that of KHAOS in several places on the curve, which has the impact of violating the social constraints as explained in the previous section. The duration of the trajectory generated by KDTP is more than 250s, far too long for a distance of a few metres. Ruckig seems to do better with a trajectory lasting about 45s but still too high. KHAOS's speed is not exceeded by Ruckig's for most of the trajectory except during deceleration where it is permanently above. Compared to KDTP, Ruckig seems to be able to generate a portion of the trajectory where the speed is stabilised at the maximum speed limit between 31 s and 36 s.

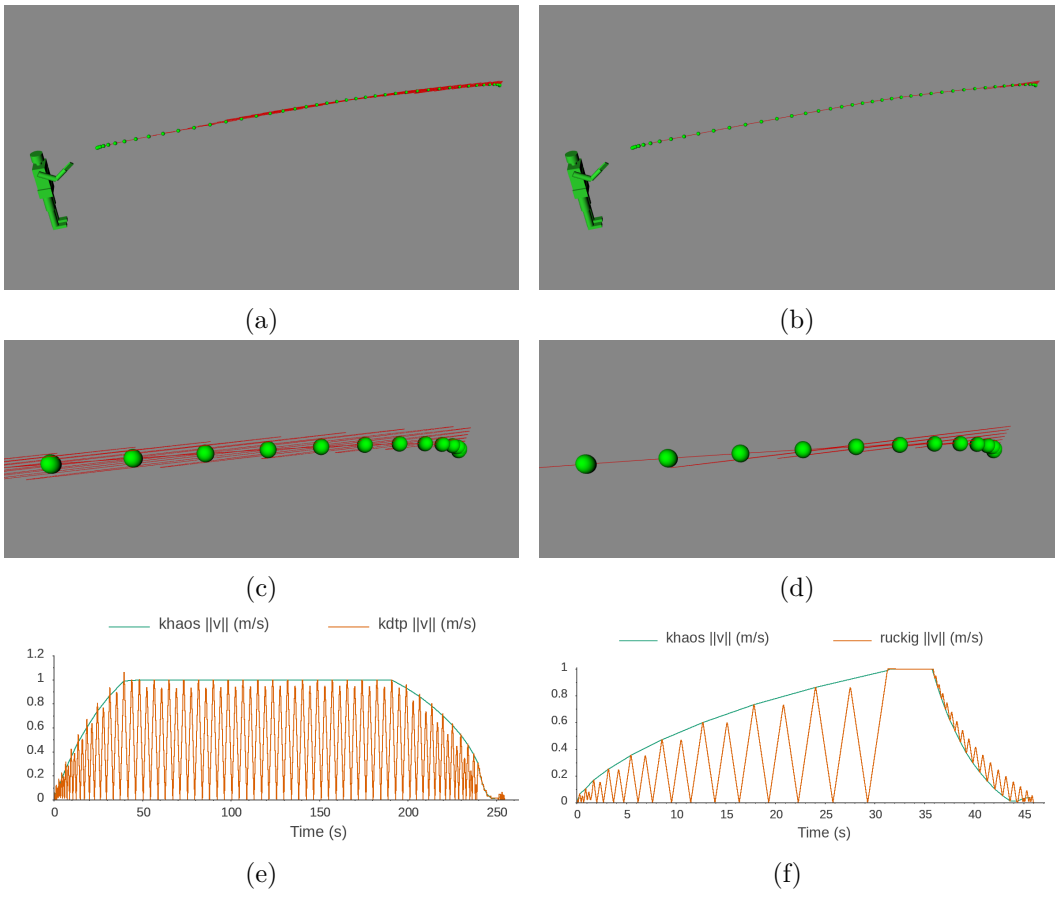


Figure 3.2: **Straigth line trajectory:** On the left (resp. right): KDTP (resp. Ruckig) results with full shape trajectory representation in a) (resp. b)). Generated trajectory is represented in red and green spheres represents the waypoints generated by KHAOS. A zoom on the start of the trajectory in c) (resp. d)) and in e) (resp. f)) the evolution of the speed magnitude as a function of time along the generated trajectory in orange with the KHAOS correspondence in green. The times on the KHAOS curves are adapted to the times of the generated trajectory which do not correspond to the real times calculated by KHAOS.

Curved trajectory The results for the curved trajectory can be seen in Fig. 3.3 which is organised in the same way as the previous figure. The trajectory overviews (Fig. 3.3a and 3.3b) again show significant trajectory deformations around the waypoints. For KDTP (Fig. 3.3a), these deformations seem to be spread all over the trajectory. Ruckig (Fig. 3.3b) seems to fit better as the oscillations are visible at the beginning, at the end and at the point where the turn is most important. For the rest of the trajectory, it seems to pass through the waypoints without deviating. Zooming in on the start of the trajectory (Fig. 3.3c and 3.3d) allows the oscillations to be better distinguished. It seems that in both cases the trajectory generator passes the waypoint and loops back to the same waypoint, as if the time or distance between the waypoints is not sufficient. Speed magnitude curves (Fig. 3.3e and 3.3f) are not very different from the straight line case with still as many oscillations and a longer trajectory duration. Ruckig still does better than KDTP but seems to be very much affected by the turn in the middle of the trajectory which is relatively simple to execute. This can be seen in the trajectory duration which is around 100 s compared to the 45 s achieved in the straight line case.

Basic trajectory The previous results show that the two trajectory generators used do not meet our needs in their current state. They also do not allow us to understand in detail what is problematic in our use of trajectory generators. We know that they need the speed and acceleration vectors as input. We also know that KHAOS does not accurately determine the components of these two vectors because the model used is very simplistic. As both libraries seem to have similar problems, we focus only on the KDTP library in the following section.

To understand a little better what is wrong, we propose below a study of a very simple and basic case. We propose to reduce the previous three-dimensional problem to a one-dimensional case. By doing so, the problem of the components of the vectors is eliminated since they are necessarily oriented in one direction only. We also reduce the difficulty by considering only two waypoints. Moreover, the path between these two waypoints is travelled at constant speed. The acceleration at these two waypoints is therefore zero. The only parameter left to study is the distance between these two waypoints.

We show Fig. 3.4 the results of KDTP under the conditions described above for three different distances between waypoints. Let's first look at Fig. 3.4a which corresponds to a distance of 2.3 m between the two waypoints. The acceleration is zero throughout the trajectory and the speed remains constant at 1m/s. The position increases linearly without fluctuation. These curves therefore correspond to the expected normal result. However, when the distance between waypoints is reduced to 2.2 m (Fig. 3.4b), unexpected deviations start to be observed. The position no longer increases linearly and the speed reaches a zero value in the middle of the trajectory before increasing again and reaching the maximum value at the arrival waypoint. The first half of the trajectory is made with negative acceleration and the second half with positive acceleration. If we reduce the distance between

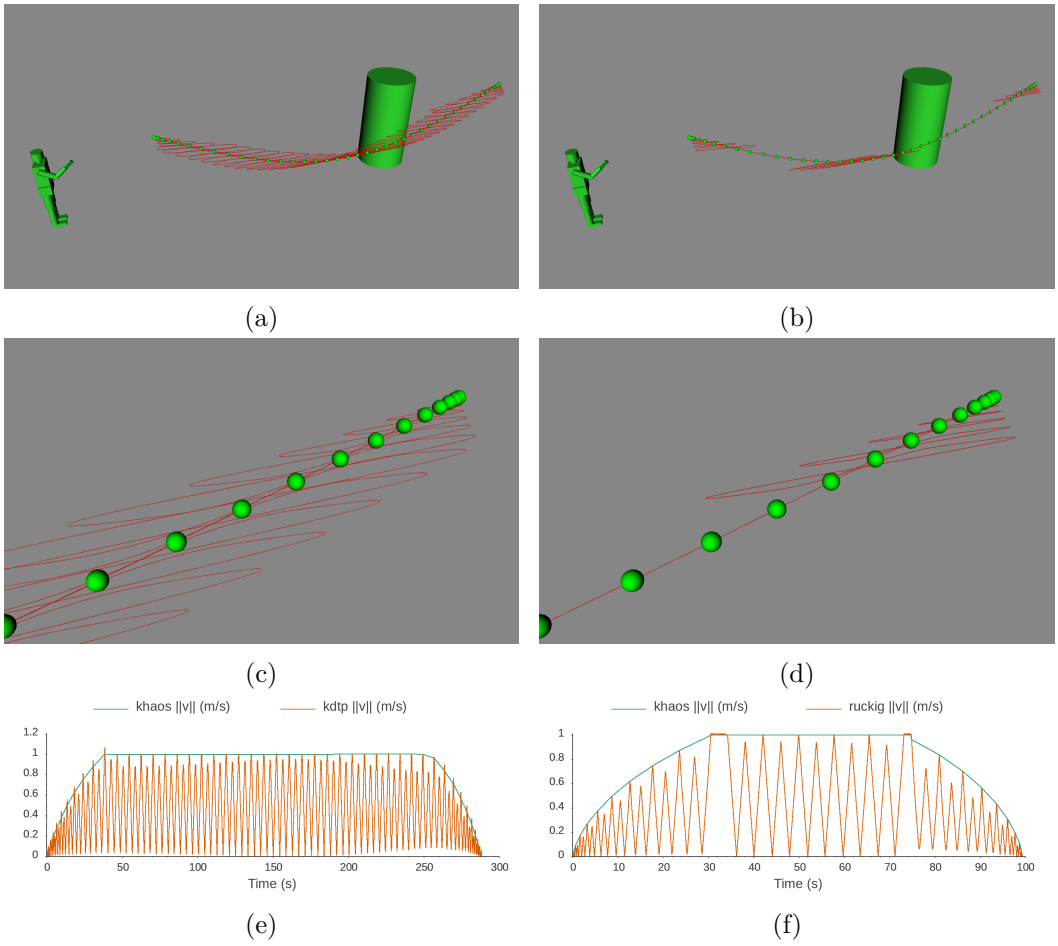
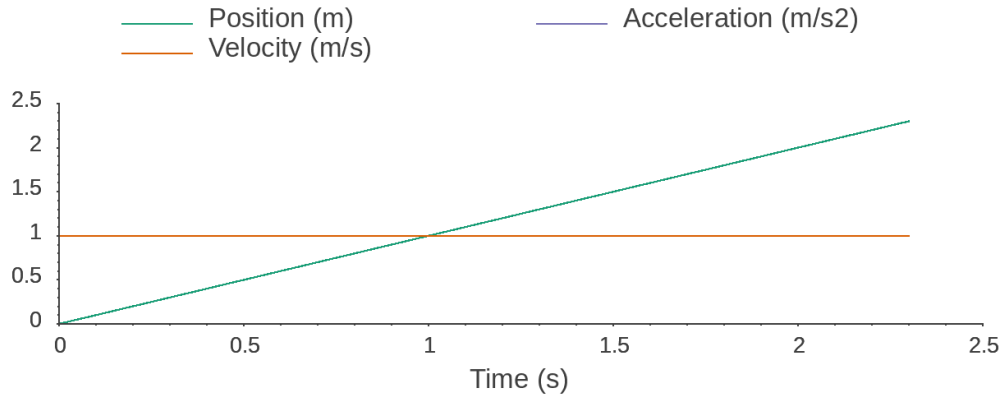


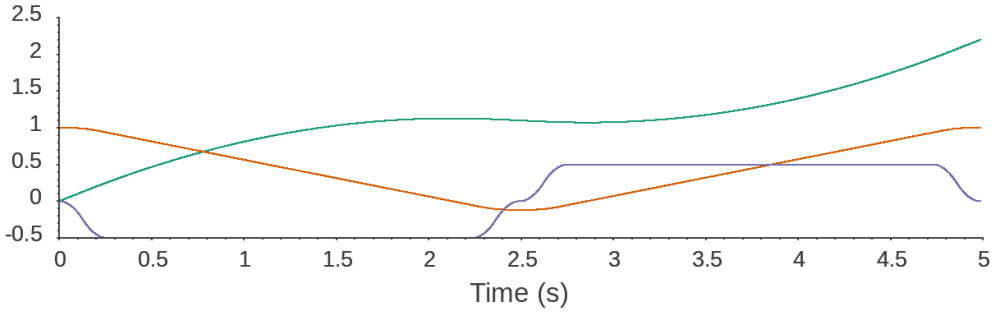
Figure 3.3: **Curved trajectory:** On the left (resp. right): KDTP (resp. Ruckig) results with full shape trajectory representation in a) (resp. b)). Generated trajectory is represented in red and green spheres represents the waypoints generated by KHAOS. A zoom on the start of the trajectory in c) (resp. d)) and in e) (resp. f)) the evolution of the speed magnitude as a function of time along the generated trajectory in orange with the KHAOS correspondence in green. The times on the KHAOS curves are adapted to the times of the generated trajectory which do not correspond to the real times calculated by KHAOS.

waypoints to 0.5 m (Fig. 3.4c), we notice the same phenomena which are amplified. The speed becomes negative and the position oscillates, representative of a backward movement as in the loops we have seen in the first figures of this section.

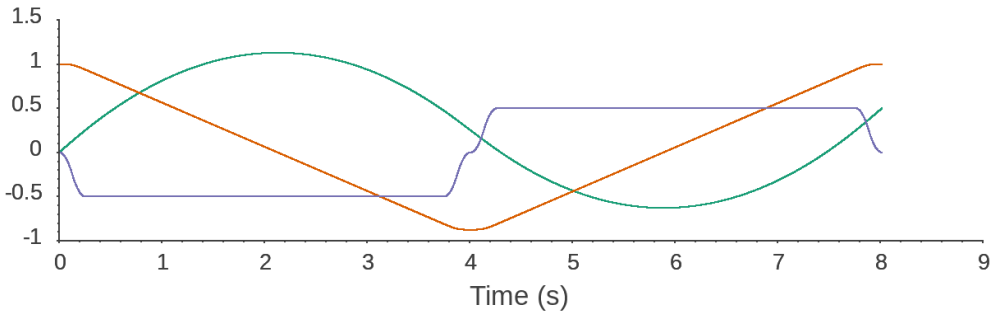
The distances between waypoints generated by KHAOS during the first tests are of the order of a few centimeters to a few tens of centimeters. We are therefore in a case similar to that shown in Fig. 3.4c. There seems to be a threshold distance between waypoints where the trajectory generator starts to encounter difficulties in adapting the speeds and accelerations to connect the two waypoints in a minimum time and in a coherent way. Under the conditions used, a minimum distance of



(a) Distance between waypoints = 2.3 m



(b) Distance between waypoints = 2.2 m

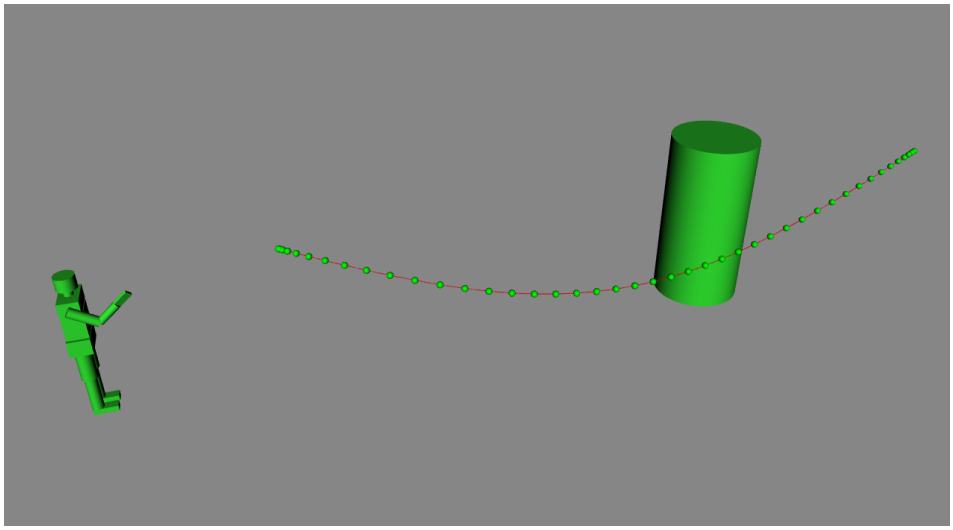


(c) Distance between waypoints = 0.5 m

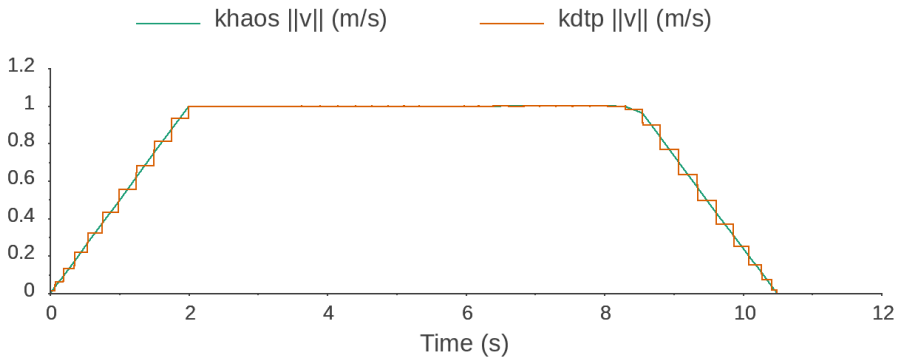
Figure 3.4: Position, speed and acceleration over time for three different distances between waypoints using KDTP library. Kinematic limits used: $|v_{max}| = 1\,\mathrm{m\,s^{-1}}$, $|acc_{max}| = 0.5\,\mathrm{m\,s^{-2}}$, $|jerk_{max}| = 10\,\mathrm{m\,s^{-3}}$ and $|snap_{max}| = 30\,\mathrm{m\,s^{-4}}$.

2.3 m between waypoints is required to obtain a satisfactory result. This solution is not feasible because such distances between waypoints prevent the trajectory generated by KHAOS from being optimised efficiently. As the waypoints serve as control points, the various costs used by the optimiser could not be evaluated at enough locations. Furthermore, if the distance between the target and the current robot position is less than 2.3 m, we cannot use our planner, which is not acceptable.

We have studied a case where the modifiable parameters are minimal. The previously mentioned threshold distance must be reducible. The only parameters that we did not vary in our study are the kinematic limits imposed on the trajectory generator.



(a)



(b)

Figure 3.5: **Curved trajectory using new kinematic limits:** KDTP result with full shape trajectory representation in a). Generated trajectory is represented in red and green spheres represents the waypoints generated by KHAOS. b) Speed magnitude as a function of time along the generated trajectory in orange with the KHAOS correspondence in green. The times on the KHAOS curves are adapted to the times of the generated trajectory which do not correspond to the real times calculated by KHAOS. Kinematic limits used: $|v_{max}| = 1\,\mathrm{m\,s^{-1}}$, $|acc_{max}| = 50\,\mathrm{m\,s^{-2}}$, $|jerk_{max}| = 10^5\,\mathrm{m\,s^{-3}}$ and $|snap_{max}| = 10^7\,\mathrm{m\,s^{-4}}$.

We show Fig. 3.5 the KDTP results for the example of the curved trajectory used earlier but with different kinematic limits. We have freed the trajectory generator by greatly increasing the imposed kinematic limits. By increasing the acceleration, jerk and snap limits this reduces the threshold distance between waypoints at which

we encountered fluctuations in the previous examples. With the new limit values, the threshold distance could be reduced to a few centimeters. This small distance makes it possible to use KDTP applied to waypoints generated by KHAOS.

We can see that the shape of the generated trajectory shown in Fig. 3.5a is suitable and does not show any loops or other visible distortions. On the other hand, if we look at the speed profile of the trajectory generated by KDTP (Fig. 3.5b), we notice sudden variations during the acceleration and deceleration phases. This can be easily explained by the very high acceleration limit applied. The speed goes from its minimum to its maximum value in a very short time, which causes the sudden variations visible on the curve. The problem of the shape of the trajectory is therefore solved, but there are still these abrupt variations which are unacceptable for our use case. We want to be able to impose limits on the trajectory generator that correspond to realistic values.

The phenomena encountered when distances between waypoints are short is a known problem. Under these conditions, it is likely that the next waypoint cannot be reached directly at the desired speeds and accelerations while respecting the kinematic constraints. The speed profile in this situation shows a slowing down before accelerating again, which is not desirable here [Sidobre 2019].

In order to bypass these difficulties, we propose in the next section to generate B-Spline trajectories that pass through the waypoints but without imposing speed and acceleration. Only the times or dates of passage at each waypoint will be used to define the kinematics.

3.4 KHAOS improvements using B-Spline

Many path planners (e.g. RRT, PRM) generate a list of waypoints consisting of at least one position information. This list is then converted into a trajectory by a trajectory generator and then into control signals which are processed and executed by a low level controller. The quality of the control signals sent to the various robot components is therefore directly linked to the quality of the trajectory effectively followed.

The trajectory generator is therefore responsible for providing a quality trajectory that takes into account the kinematic limits of the robot that must execute it. The quality of a trajectory is characterised by the necessity to respect numerous constraints. Firstly the smoothness of the input trajectory of a controller is important to guaranty good following properties. In our case we consider essentially trajectories defined by cubic functions over time for which the second derivative or acceleration is continuous. Secondly the controller and the system impose constraints limiting the velocity, the acceleration and the third derivative or jerk. Finally the context of the task to be realised and in particularly the humans introduces new constraints in the operational space. The latter constraints also limit the geometric and kinematic characteristics of the trajectory.

Most of these trajectory generators use various mathematical functions such as

cubic functions or splines to connect waypoints. If the distances or times between each waypoint are large, the mobile has time to accelerate and decelerate to reach the waypoints on the planned dates. It is therefore relatively easy for the trajectory generator to find a curve with good parameters that ensures continuity. On the other hand, if the distances or times are short, we may encounter the phenomena presented in the previous section. Unintentional spatial deformations of the trajectory or strong variations in acceleration may occur.

Moreover, KHAOS generates a list of waypoints composed of position, speed and time information in a context of Human Robot Interaction. In order not to violate the social constraints planned by KHAOS, we need the spatial deformation between two waypoints to be minimal and the speeds to be well respected. It is therefore crucial that the trajectory generated by the trajectory generator faithfully follows the planned positions and respects the times or dates imposed on each waypoint.

Generating a time-continuous trajectory from a set of waypoints with given dates and kinematic constraints remains an open problem. To this end, after a short introduction to the Non-Uniform Cubic B-Spline we propose in this section the use of a new extension of the SoftMotion library and in particular the use of Non-Uniform Cubic B-Spline [Piegl 1996] to generate feasible trajectories from a list of waypoints.

3.4.1 Introduction to B-Spline

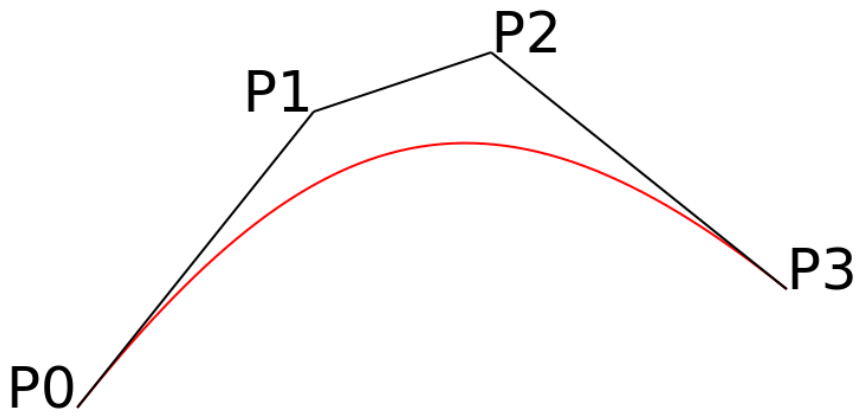


Figure 3.6: Cubic Bézier curve (n=3) representation (in red) with the corresponding Bézier control polygon in black

Bézier curve In order to understand what a Non-Uniform Cubic B-Spline is, let us first give a short introduction to what a Bézier curve is. A Bézier curve composed of $n+1$ control points $(P_0, \cdots, P_n)$ is the set of points defined by:

$$P(u) = \sum_{i=0}^{n} B_i^n(u) \cdot P_i \quad \text{with} \quad u \in [0;1] \tag{3.1}$$

where $B_i^n(u)$ are the Bernstein polynomials [Bernstein 1912] and $(P_0, \cdots, P_n)$ are the control points forming the "Bezier control polygon".

This definition of a curve can be assimilated to a trajectory by replacing the parameter u where $u \in [0;1]$ by time t where $t_{min} \leqslant t \leqslant t_{max}$. From now on we will only talk about trajectories.

One of the properties of the Bernstein polynomials B_i^n is the following:

$$\sum_{i=0}^{n} B_i^n(t) = 1 \tag{3.2}$$

So for $n = 3$ we can write:

$$1^3 = (1 - t + t)^3 = ((1-t)+t)^3 = (1-t)^3 + 3(1-t)^2 t + 3(1-t)t^2 + t^3 \tag{3.3}$$

Including the $n + 1 = 4$ control points P_i we get:

$$P(t) = B_0^3(t) * P_0 + B_1^3(t) * P1 + B_2^3(t) * P2 + B_3^3(t) * P3 \quad \text{with} \quad t \in [t_{min}; t_{max}]$$
$$B_0^3(t) = (1-t)^3$$
$$B_1^3(t) = 3(1-t)^2 t$$
$$B_2^3(t) = 3(1-t)t^2$$
$$B_3^3(t) = t^3$$

This gives us the Bézier curve of degree 3, also known as the cubic Bézier curve, which is composed of four control points. Figure 3.6 shows an example of a cubic Bézier curve and the "Bézier control polygon" formed by the four control points $(P_0, \cdots, P_3)$. It can be seen that the curve necessarily passes through the control points P_0 and P_3. However, it does not necessarily pass through the other control points P_1 and P_2. A Bézier curve is infinitely differentiable (of class $\mathcal{C}^\infty$), and the vector $\overrightarrow{P_{n-1}P_n}$ is the tangent vector at point P_n.

Non-Uniform Cubic B-Spline Bézier curves characteristics presented in the previous paragraph now allow us to introduce the Non-Uniform Cubic B-Spline. A pth-degree B-Spline curve is defined by:

$$C(t) = \sum_{i=0}^{n} N_{i,p}(t) \cdot P_i \quad \text{with} \quad t_{min} \leqslant t \leqslant t_{max}$$
$$U = (t_{min}, \cdots, t_{min}, t_{p+1}, \cdots, t_{m-p-1}, t_{max}, \cdots, t_{max})$$

where the P_i are the control points, and the $N_{i,p}(t)$ are the pth-degree B-Spline basis functions defined on the non uniform knot vector U that includes $m+1$ knots t_i.

One way to define the B-Spline basic functions is to use a recurrence formula due to Carl de Boor [De Boor 1972]. The ith basis B-Spline function of p-degree (order $p+1$), denoted by $N_{i,p}(t)$ is defined as:

$$\begin{aligned} N_{i,0}(t) &= \begin{cases} 1 & \text{if} \quad t_i \leqslant t < t_{i+1} \\ 0 & \text{otherwise} \end{cases} \\ N_{i,p}(t) &= \frac{t - t_i}{t_{i+p} - t_i} N_{i,p-1}(t) + \frac{t_{i+p+1} - t}{t_{i+p+1} - t_{i+1}} N_{i+1,p-1}(t) \end{aligned} \tag{3.4}$$

We have deliberately named the knots t_i because for our use of the B-Spline we can consider the knots as the times or dates of our trajectory. This is better adapted to consider waypoints where knots defines the instant when the trajectory passes through the waypoint. Since the time between each waypoint is not necessarily the same, we use the generic form of B-Spline which is called Non-Uniform Cubic B-Spline which have the particularity to accept different gaps between knots.

As we have seen, the Non-Uniform Cubic B-Spline do not necessarily pass through the control points. The next section introduces the use of a new extension to the softMotion library by describing how it handles the correspondence between the control points and the waypoint positions of the trajectory given by KHAOS.

3.4.2 SoftMotion extension using Non-Uniform Cubic B-Spline

SoftMotion library SoftMotion[1] is a library of functions for planning, generating, controlling and manipulating trajectories in the context of HRI. The first tool was introduced in 2008 by Broquere et al. [Broquere 2008], it is a one-dimensional time optimal trajectory generator that produces trajectories satisfying symmetric bounded speed, acceleration and jerk limits. These trajectories are expressed as a sequence of trajectory segments defined by cubic polynomial functions over time. The work has been extended to the N-dimensional case and integrated into a complete planning and control system [Broquere 2010]. Later a complete algorithm for the one-dimensional time optimal trajectory generator with velocity, acceleration and jerk asymmetric bounds is described in [Sidobre 2019]. This work is extended by Desormeaux et al.[Desormeaux 2019] to bring back the current state of the robot inside the admissible domain when one or more constraints are exceeded or violated. This occurs in particular when limits are reduced, for example when the maximum speed is reduced to ensure the safety of a human approaching the robot.

SoftMotion extension To meet our needs, we propose in this section to use a new extension to SoftMotion based on Rousseau's thesis work [Rousseau 2019]. In his work, one of the goals is to follow a cinematographic flight plan using a

[1] https://git.openrobots.org/projects/softmotion

quadrotor. Just as in our use case, it is necessary to generate a smooth trajectory passing through a set of waypoints. In his case, smoothing is essential in order to obtain sufficient stability for taking pictures in mid-air. To do this, he uses B-Spline to generate a minimum time trajectory constrained by the speeds, accelerations and jerk.

In this new SoftMotion extension, trajectories are generated using Non-Uniform Cubic B-Spline constrained by the positions and speeds at each waypoint provided by the output of our KHAOS planning system. Global kinematic limits in acceleration and jerk are also applied to control the behaviour of the trajectory.

As described earlier, a Bézier curve or a B-Spline does not necessarily pass through its control points. It is therefore an approximation of these control points that is carried out except at the end control points through which the curve necessarily passes.

In our use case, we want the curve to faithfully connect the S_i positions of the waypoint list given by our KHAOS planner. We therefore want to interpolate the S_i with a B-Spline curve. Let be a Non-Uniform Cubic B-Spline defined by the control points P_i and the instants or knots t_i : At t_i it approximates the point P_i and passes through S_i. We therefore know a list of waypoints S_i from which we will calculate the control points P_i.

By developing the terms of equation 3.4 in the case of an Non-Uniform Cubic B-Spline, i.e. $p = 3$, we see that many terms are zero allowing the equation of the curve to be simplified.. At time t_i the trajectory reaches the waypoint S_i, which can be expressed as :

$$C(t_i) = S_i = a_i P_{i-1} + b_i P_i + c_i P_{i+1} \tag{3.5}$$

where $a_i = N_{i,3}(t_i)$, $b_i = N_{i+1,3}(t_i)$ and $c_i = N_{i+2,3}(t_i)$ are the only three non-zero coefficients.

To control the velocities and accelerations at the beginning and end of the trajectory, some points are added to the list of points to be interpolated. These points as well as the associated knots are chosen on a constant acceleration trajectory segment built to extend the trajectory in a continuous way. The trajectory is then truncated to keep only the useful part.

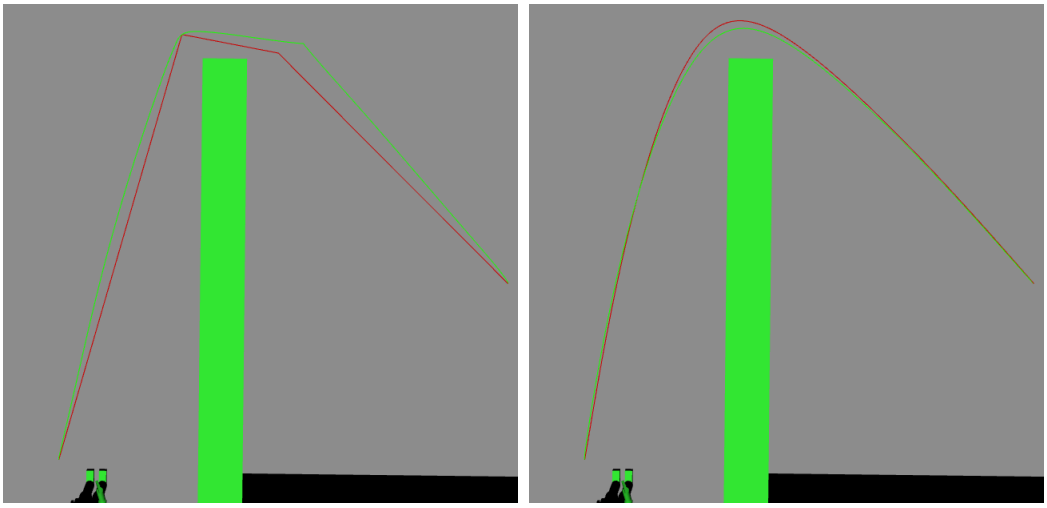
The set of these equations defines a tridiagonal matrix which is solved in N complexity where N is the number of points to interpolate. By solving this system of equations, we obtain the correspondence between the desired positions at each waypoint of the trajectory and the control points of the Non-Uniform Cubic B-Spline. Secondly, the functions already integrated in softMotion can be used to determine the kinematic components along the trajectory.

The main advantage of this extension is to allow the control of the kinematic parameters at the beginning and at the end of the trajectory. This is necessary to ensure continuity between consecutive trajectories generated by two successive plannings as in our case of reactive planning in a human-populated environment.

3.5 Results

In this section, we show the results of the trajectory generator using Non-Uniform Cubic B-Spline described earlier. We use it at two stage: first to smooth the initial path and then to produce the final trajectory. Firstly we show the improvement in the quality of the initial path that is essential for the proper functioning of our planner. We then take the examples presented in section 3.3 and show the improvements introduced by the use of this trajectory generator based on BSpline. Finally, we introduce a new situation where two humans are present to show the evolution of the speed profile during the execution of the trajectory.

3.5.1 Initial path smoothing



(a) Standard KHAOS (b) KHAOS using BSpline

Figure 3.7: Shape of the initial path (red) and trajectory generated by KHAOS (green) after several iterations. a) No smoothing is applied b) BSpline smoothing is applied

The use of BSpline allow to generate a smooth trajectory from a list of waypoints even when they are numerous and very close. Many planners such as Probabilistic RoadMap (PRM) or Rapidly exploring Random Tree (RRT) allow to generate an initial path connecting a start point and an end point using a list of waypoints. Between each waypoint composing this initial path, we have segments that by nature do not allow to respect a continuity of speed. As detailed at the beginning of this chapter, this lack of continuity can be reflected in the output of KHAOS and make its optimisation phase inefficient.

We compare Fig.3.7, the outputs of the standard version of KHAOS when no smoothing is applied (Fig.3.7a) and when a B-Spline is used (Fig.3.7b). When no smoothing is performed, we show that even after many iterations, the continuity defects present in the initial path are transmitted to the KHAOS result as presented

in 3.2. On the other hand, when using B-Spline, it can be seen that the initial path represented in red is smooth and has no sharp breaks. We can also observe that the shape of the trajectory generated by KHAOS after several iterations is also very smooth.

This result shows the influence of the quality of the initial path on the result of the planner. The smoothing applied on the input side alone allows to considerably improve the shape of the trajectory on the output side of KHAOS. This provides smoothness which also facilitates continuity in kinematics as presented in the next section.

3.5.2 Kinematic compliance

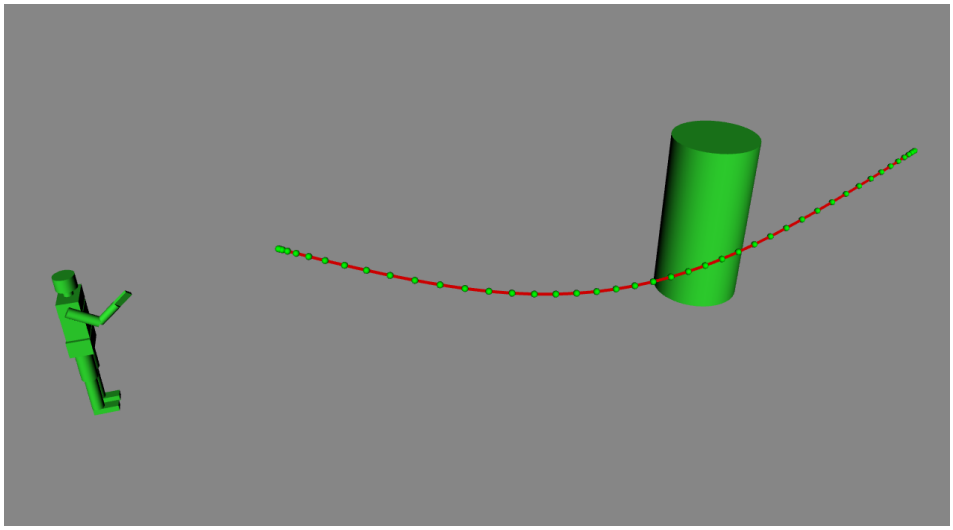
Now that the initial path smoothing defect is resolved, we can tackle the kinematic compliance along the trajectory detailed at the beginning of this chapter.

The previous example showed the smoothing of the initial path consisting of only four waypoints including the start and the goal. The distances between waypoints are therefore relatively high. It is necessary to test with a larger number of waypoints more or less close together. If this test is successful, i.e. the shape of the trajectory is continuous without distortion, it remains to be verified that from a kinematic point of view the speed profile is also smoothed.

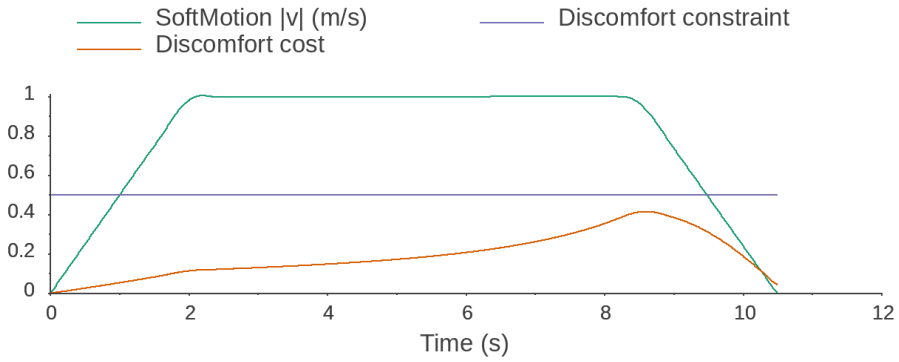
We propose in a first step to use the same examples of "Curved" and "Straigth line" trajectories as presented in section 3.3 where the trajectory generators KDTP and Ruckig encounter difficulties. The kinematic limits used are the same as those used by KHAOS and are indicated in the legends of each figure. In order to verify the correct functioning of our B-Spline solution, we are also interested in the evolution of the *discomfort_cost* and the respect of the *discomfort_constraint* presented in 2.3.2.1.

Curved trajectory using BSpline In this paragraph we take up the example of the curved trajectory presented earlier where the trajectory is deviated by an obstacle and approaches a human. The result using BSpline is shown in Fig. 3.8. The waypoints of the curved trajectory generated by KHAOS are represented by green spheres in Fig. 3.8a. The result given by the BSpline is also represented by a red curve. We notice that the trajectory is well smoothed and passes through all waypoints without any distortion.

We also show the evolution of the *discomfort_cost* in parallel with the speed profile in Fig. 3.8b. The applied *discomfort_constraint* is also shown to provide a visual cue on the graph. The value of the *discomfort_constraint* is set to 0.5 in this example. First of all, we can see that the speed is not distorted and appears to be correctly smoothed. The speed limit of $1\,\mathrm{m\,s^{-1}}$ is not exceeded except towards the end of the acceleration phase where a slight jump can be noticed. The evolution of the *discomfort_cost* is also smooth and representative of a smooth movement of the Flying CoWorker if it were to execute this trajectory. Finally, we note that the *discomfort_constraint* is respected at all times, particularly when approaching the



(a)



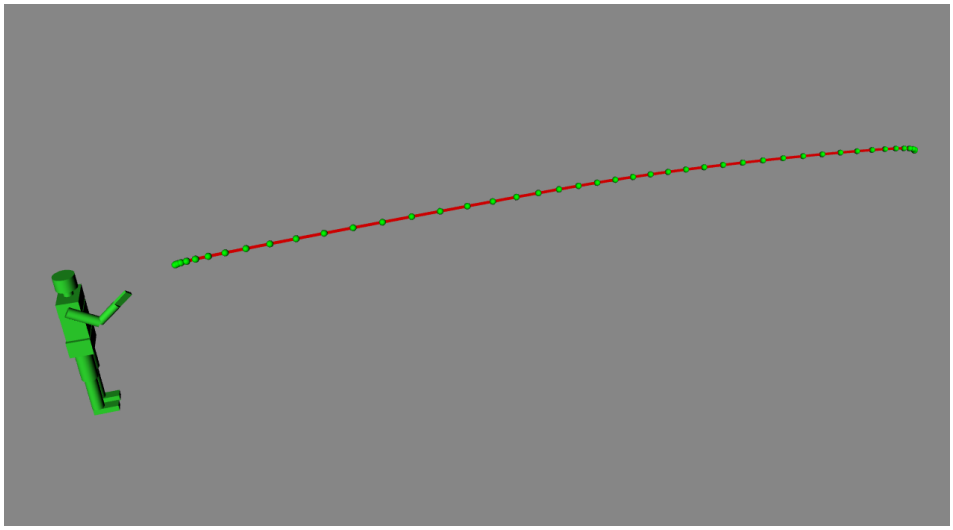
(b)

Figure 3.8: **Curved trajectory using BSpline:** a) full shape trajectory representation. Generated trajectory is represented in red and green spheres represents the waypoints generated by KHAOS. b) Speed magnitude, *discomfort_cost* as a function of time along the generated trajectory for a *discomfort_constraint* of 0.5. Kinematic limits used: $|v_{max}| = 1\,\mathrm{m\,s^{-1}}$, $|acc_{max}| = 0.5\,\mathrm{m\,s^{-2}}$, $|jerk_{max}| = 10\,\mathrm{m\,s^{-3}}$.

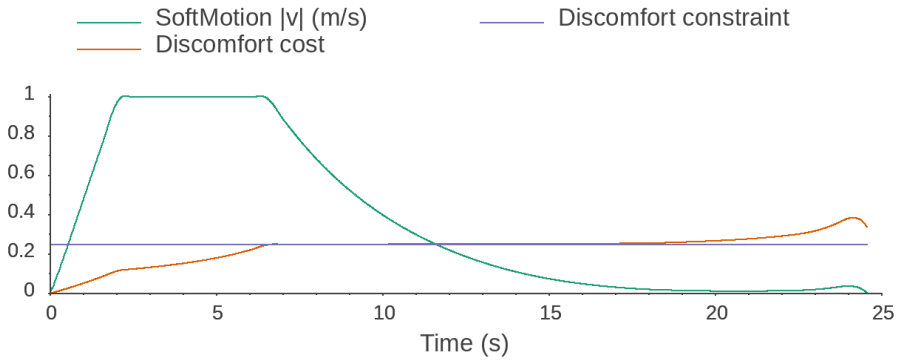
human. In this example, the *discomfort_constraint* is rather high and is therefore not very restrictive, especially as the objective of the trajectory is quite far from the human.

Straigth line trajectory using BSpline The example of the straight line trajectory shown in Fig. 3.9 further challenges the trajectory generator at the level of social constraints. Indeed, the goal of the trajectory is very close to the human compared to the previous example. Moreover, the chosen discomfort constraint is only 0.25. As a result, the trajectory planned by KHAOS forces the Flying CoWorker to decelerate very early and to move at a very slow speed close to the human.

The shape of the generated trajectory using BSpline is shown in Fig. 3.9a. Again,



(a)



(b)

Figure 3.9: **Straigth line trajectory using BSpline:** a) full shape trajectory representation. Generated trajectory is represented in red and green spheres represents the waypoints generated by KHAOS. b) Speed magnitude, *discomfort_cost* as a function of time along the generated trajectory for a *discomfort_constraint* of 0.25. Kinematic limits used: $|v_{max}| = 1\,\mathrm{m\,s^{-1}}$, $|acc_{max}| = 0.5\,\mathrm{m\,s^{-2}}$, $|jerk_{max}| = 10\,\mathrm{m\,s^{-3}}$.

it can be seen that the trajectory generator faithfully follows the result planned by KHAOS without distortion between waypoints. As in the previous example, we represent Fig. 3.9b the evolution of the speed magnitude and the *discomfort_cost* by indicating with a horizontal line where the *discomfort_constraint* is located. This shows that the speed profile is quite correct on the first part of the trajectory. When the discomfort constraint of 0.25 is reached, the deceleration phase starts as the FCW continues to approach the human and therefore has to adapt its speed by reducing it. The BSpline produces a very smooth deceleration without distortion. However, at the end of the deceleration phase, when the Flying CoWorker is close to the human, the *discomfort_constraint* is exceeded in the last few seconds.

This fault is not critical because between 16 s and the end of the trajectory, the Flying CoWorker moves at a speed lower than $0.05\,\mathrm{m\,s^{-1}}$. As the system is highly constrained in this example, the slightest deviation in this speed range can quickly lead to constraint violation.

The reason for not respecting the *discomfort_constraint* in this particular case does not come from the trajectory generator itself but rather from a wrong evaluation of the deceleration by KHAOS. Indeed, we recall that KHAOS uses a basic constant acceleration model for the calculation of speeds. It is therefore possible that in cases requiring precision like this one, the KHAOS speed evaluation model is not sufficient. This is because the trajectory generator is not aware of the *discomfort_cost* or *discomfort_constraint*, it only relies on the speeds and positions given by KHAOS and tries to follow these data as reliably as possible.

3.6 Conclusion

In this chapter we have proposed a solution based on the use of B-Spline to improve the behaviour of the Flying CoWorker using our planner KHAOS presented in the previous chapter.

First, we have shown the importance of improving the initial path given as input to our planning system. Then we have shown, with the help of two trajectory generators present in the literature, the difficulties to generate a continuous-time trajectory from the list of waypoints given by KHAOS. We have identified that the small distance between the waypoints generated by KHAOS and the need to faithfully respect the constraints at each waypoint make it difficult to convert to a time-continuous trajectory.

Following this, we proposed the use of a new extension of the SoftMotion library using B-Spline. We have shown the interest of using the mathematical properties of BSplines and especially their ability to respect a continuity. Finally, we have shown through different examples that the use of B-Spline can considerably improve the input and output of KHAOS. We have thus improved the smoothing of the initial input path of KHAOS, preventing the optimizer from repeating unintended faults. In addition, we now have a way to convert a KHAOS trajectory into a time-continuous trajectory. The particularity of this trajectory is to respect the social constraints planned by our system while ensuring a feasible and executable solution from a kinematic point of view. This particularity allows this trajectory to be executed by a low level controller.

Chapter 4

Human-FCW Handover planning and control

Contents

4.1 Introduction 57
4.2 Flying CoWorker design and handover 58
4.3 Related work 60
4.4 Flying CoWorker base orientation for Handover 62
4.4.1 Orientation calculation details 63
4.4.2 Orientation behavior during approach 63
4.4.3 Discussion 65
4.5 Reactive FCW goal state estimation for handover 66
4.5.1 Context 66
4.5.2 Reactive goal estimation 68
4.6 Coordinated motion for FCW 79
4.6.1 Context 80
4.6.2 KHAOS main algorithm extension for coordinated motion . . 80
4.6.3 Coordinated handover results 87
4.7 Conclusion 95

4.1 Introduction

We now know how to generate a trajectory for the Flying CoWorker base using our Kinematic Human Aware Optimization-based System for reactive planning of flying-coworker presented in Chapter 2. This trajectory allows the Flying CoWorker to navigate in an environment populated with humans and various dynamic obstacles. It considers the kinematic constraints of the Flying CoWorker but also social constraints such as the visual field and the discomfort generated to the human. We know that these constraints are valid in the proximity of humans and can therefore be used for a handover task. We know that abrupt motions or interruptions in a robot's motion can be uncomfortable for the human and can affect the human's understanding of its intention. We therefore want a system that offers the smoothest possible motion.

In this chapter, we address the handover problem based on our planner KHAOS which, in its initial version, provides many advantages for evolving in proximity to humans. We first propose a model of Flying CoWorker which is used in the rest of this chapter. After the state of the art, we present a method to determine the orientation of the Flying CoWorker base specific to a handover task. We continue by presenting a method to determine the final complete state of the Flying CoWorker for a handover task. Then, we propose an extension of KHAOS to generate the complete trajectory of the Flying CoWorker with the particularity of coordinating the motion of its arm and base. Finally, we present results of trajectories generated with this extension to show the capabilities of our planner in simulation. To do this, we use the human model and some of the situations used in the previous chapters.

4.2 Flying CoWorker design and handover

Discussion on the FCW design In order to better demonstrate the possibilities of the proposed planning system, we propose in the remainder of this manuscript to work only in simulation using the Jaco arm model from KinovaRobotics [Campeau-Lecours 2019] as a replacement for the 3 degrees of freedom arm currently fitted to the Flying CoWorker platform. Jaco's arm design offers 6 degrees of freedom allowing a greater number of configurations and whose length is also greater with a maximum reach of 0.9 m. The design and construction of such a lightweight 6DOF arm for the Flying CoWorker application would be a significant challenge, but it could allow us to tackle more complex HRI problems as the various examples in this chapter demonstrate.

Indeed, the initial Flying CoWorker platform is equipped with a manipulator arm with 3 degrees of freedom as described in 1.2. It is build mainly in carbon composite using 3D printed parts for supports and joining parts. The diameter of the base approaches 1 m when considering the propellers while the arm is just over 0.5 m. This means that even at full extension of the arm, it is difficult for a human to perform a handover with the Flying CoWorker without being very close to the propellers, especially if the object to be transferred is small.

The base used is a hexarotor with tilted propellers which provides very good stability in flight. The orientation of the base can be defined by 3 angles Yaw, Pitch and Roll. Unfortunately, the maximum tilt for this design is about 10° which limits Pitch and Roll. As this tilt angle is relatively small, it does not allow for real consideration at planning level. We prefer to reserve this freedom of motion for control in order to ensure better stability. Therefore, we simplify the planning level by assuming these two angles to be fixed and by acting only on the yaw angle.

Also, increasing the length of the arm would require increasing its weight, decreasing the payload of the Flying CoWorker and potentially increasing the diameter of the base to increase the thrust required for flight. A possible hardware solution would be to tilt the holding bars connecting the propellers to the centre of the base in such a way that the propellers would be at a much higher altitude than the

base. This would increase the accessible area under the Flying CoWorker but would require significant changes in the design and control laws of the current platform.

Proposed model The model used in the remainder of this chapter is illustrated in Fig. 4.1 where the elements that make up the platform are shown. The large blue cylinder corresponds to the bounding cylinder of the hexa-rotor. The small blue, cyan, red and black bounding cylinders correspond to the parts of the kinematic chain of the arm. Finally, the purple cylinder represents an example of an object to be exchanged.

We also introduce the different frames that define the location of the Flying CoWorker joints as well as the *end_effector* holding the object to be exchanged (purple cylinder). The small black, red and blue cylinders each one actually corresponds to two links that can rotate in relation to each other along the cylinder's axis of revolution. From a collision object point of view, they can therefore be represented by a single cylinder. The small blue cylinder represents the link between the base of the arm and the shoulder that can rotate around the z_0 axis. Cyan cylinder represents the upper part of the arm linking the shoulder to the elbow. Between the elbow and the wrist is the forearm represented by the red cylinder. Finally, the black cylinder represents the link between the wrist and the hand (or *end_effector*) of the arm.

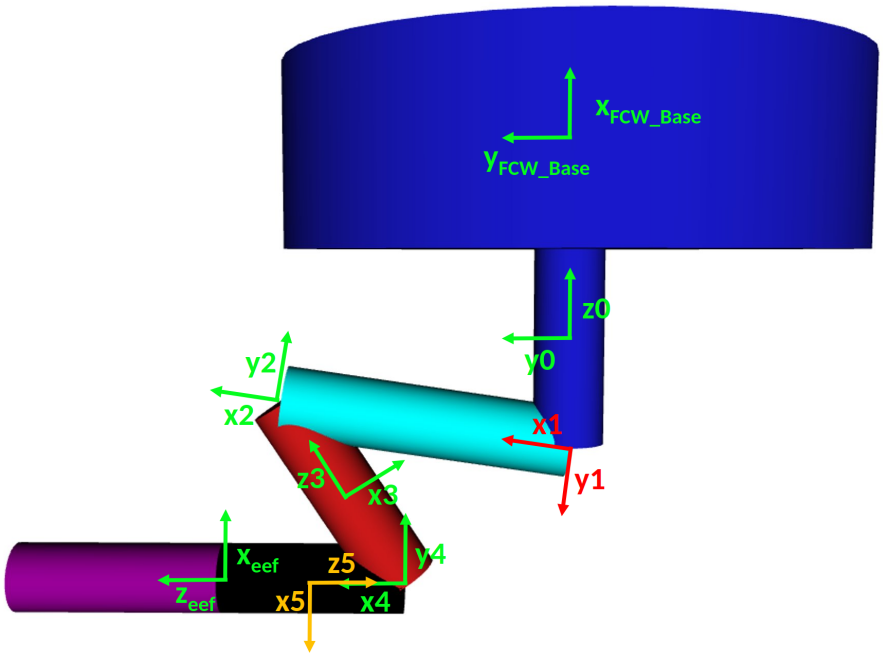


Figure 4.1: Flying CoWorker model used for collision checking and frames of the platform consisting of the hexa-rotor (large blue cylinder), the equipped arm (small blue, cyan, red and black cylinders) and an example of object to be exchanged (purple cylinder). Each reference frame represents a rotary joint about the z_i axis.

4.3 Related work

Handover is a generic term used in robotics to describe a joint action to exchange an object. In our case this action is performed between a human and a robot where each can have the role of the giver or the receiver [Fiore 2016]. But this generic term can gather a large number of phases and subjects to be studied in order to make this action happen in the best possible conditions. A recent review by Ortenzi et al. [Ortenzi 2021] on the subject of handover classifies the work into two main phases. The first is the pre-handover phase while the second concerns the physical exchange of the object.

Physical exchange concerns all the part where the physical contact takes place. The control of the exchange phase can be entirely realized by the human, for example if he/she hangs the object on a hook carried by the robot, but the objective is to realize a more natural exchange, i.e. an exchange where the robot behaves as a second human would. In this case it is necessary to detect the contact, to control the contact forces and to estimate the quality of the human's grasp as treated by He and Sidobre [He 2015] using "Bidule". If the robot has the role of giver, it must differentiate between a simple tactile effort by the human and a real grasping of the object by the human so as not to drop the object unintentionally as in the work from Medina et al. [Medina 2016]. If it is the receiver, the robot must control the effort applied to the object to remove it at the right moment and not pull the donor's arm by withdrawing too violently for example as shown by Pan et al. [Pan 2018]. To accurately detect these contacts, force sensors are used which can for example be mounted on the wrist of a manipulator arm as used by Konstantinova et al. [Konstantinova 2017]. It is common to combine these force sensors with visual feedback to predict and detect possible collisions or to anticipate the motion speed of the receiver's arm as in the work of Controzzi et al. [Controzzi 2018].

The phase dealt with in this manuscript concerns the pre-handover phase, which itself comprises many stages. Before starting the physical interaction with the human, it is necessary to determine the grasp of the object envisaged by the human, which will induce the grasp by the robot itself. This grasp depends first of all on the geometry as detailed by Miller et al. [Miller 2003] and mass of the object in accordance with the *end_effector* of the robot used [Goldfeder 2007]. Complex environments will also influence the choice of grasp especially when objects are close to obstacles like shown by Berenson et al. [Berenson 2007]. Faced with all these challenges representing an infinite number of solutions, an interesting approach consists in generating a representative list of grasp and then choosing a good one as handled by Saut et al. [Saut 2012]. Recent advances in neural networks make it possible to define the grasp of the object from a large number of data as in the work from Yang et al. [Yang 2020] and then further refined using model predictive control to achieve a smooth handover [Yang 2022].

Until now, robot grippers have been rigid with few degrees of freedom, which severely limits their ability to manipulate objects. Most of the time, robots and especially those with only one arm, cannot re-manipulate the object after grasping

it. As a result, the robot's grasp on the object during the exchange will be the same as when it started its trajectory. As the environment and the human's situation can be greatly modified during the execution of the trajectory, choosing a good grasp on the object is not sufficient or even useless in some cases because another one can quickly become more interesting. It is thus necessary to foresee the evolution of the human's posture and its close environment in time to predict a good grasp pose and location to exchange the object as explored by Li et al. [Li 2015].

Finding the right place to exchange the object can be defined in many ways which may depend on the context. Mainprice et al. [Mainprice 2012] presents one such way by describing a shared effort between a PR2 robot (giver) that depends on the *mobility* of the human (receiver), taking into account the difference between an old person sitting on a chair and a young person standing. In another paper Mainprice et al. [Mainprice 2010] proposed a sampling-based planner dealing with handover in near human space with the use of a costmap considering visibility, human's posture and safety. Several studies have looked at how to approach the human for a handover. Koay et al. [Koay 2007] concluded in their study that people generally prefer a frontal approach while in Dautenhahn et al. [Dautenhahn 2006]'s study shows a preference for a left or right approach depending on the task. The exchange area can be reduced by considering the constraints of the human arm as presented in Vianello et al. [Vianello 2021] work to avoid musculoskeletal disorders. A slightly different approach allows Rasch et al. [Rasch 2018] to apply the constraints of the human arm to the robot arm in order to achieve a more sociable behaviour. More recently, Corsini et al. [Corsini 2022] present in their work a Nonlinear Model Predictive Controller for handover considering the effort needed by the human arm and its visual field.

To guide the robot's choices for a handover and to help the human understand these choices, it is important for both to communicate their intention. Moon et al. [Moon 2014] were interested in the importance of gaze in this communication of intention and how it can affect the timing of the handover. Another study by Gharbi et al. [Gharbi 2015], referring to head motion associated with gaze, shows a preference for two behaviours, **OR** (the giver looks at the Object then at the Receiver) and **ROR** (the giver looks at the Receiver then at the Object and then at the Receiver again) for object handover. Unfortunately aerial robots have very limited means to communicate. Many signals are therefore transmitted through their motion. Some work, such as that from Cauchard et al. [Cauchard 2015], looks at the communication channels for a human to tell the drone what action to take. Szafir et al. [Szafir 2014] and more specifically Jensen et al. [Jensen 2018] have worked on the motion of the drone to indicate to the human an intention or to give information. There is obviously the motion of the robot and the human to take into account but the timing of the interaction is also essential to understand an intention as discussed in the work of Cakmak et al. [Cakmak 2011].

Naturally, a human moves and gives an object to another human in a coordinated way without distinguishing a break between the phase where he/she approaches by walking and the phase where he/she gives the object by extending his/her arm. To

our knowledge, there is no work in the Human Robot Interaction literature on the coordinated motion of an aerial robot equipped with a manipulator arm interacting with a human during a handover such as the Flying CoWorker. It is necessary to look at the fields of motion planning for mobile manipulators as presented in a review from Sandakalum et al. [Sandakalum 2022] and co-manipulation to find work relating to the generation of coordinated motion of a robot equipped with a mobile base and a manipulator arm.

In the field of co-manipulation, Peternel et al. [Peternel 2017] shows a mobile manipulator collaborating with a human to move an object taking into account ergonomic requirements for the human co-worker which requires a very complete and complex model of the human. In a completely different area, Vazquez et al. [Vazquez-Santiago 2021] plan the motion of a mobile welding robot starting from the initial welding path. Due to the precision required, the system takes several minutes to find a unique solution not allowing to work with a human. In contrast, a mobile non-holonomic manipulator is controlled using a visual servo process by Li et al. [Li 2021], the system is able to track a target by simultaneously planning for the base and the arm in real time. As part of a coordinated motion of a mobile manipulator, it is difficult to choose when it is better to move the arm or rather the base as shown by Xing et al. [Xing 2021] where several different behaviors of the robot are presented. To define a good behavior of the mobile manipulator, many works such as those of Zhang et al. [Zhang 2016] or Huang et al. [Huang 2000] are based on the concept of manipulability defined by Yoshikawa [Yoshikawa 1985]. The manipulability makes it possible to evaluate the configuration of the planned arm in order to avoid singularities from a control point of view but does not necessarily ensure a good motion in a Human Robot Interaction point of view.

After this presentation of the bibliographic context, we will extend our planner to the planning of the orientation of the Flying CoWorker base in order to communicate to the human its intention to initiate a handover.

4.4 Flying CoWorker base orientation for Handover

Before dealing with the handover problem integrating the motion of the Flying CoWorker's arm, we first want to present a way for the Flying CoWorker to signal its intention to the human using only its base. We assume that the base is equipped with a device to identify its direction of travel, such as a camera attached at the edge of the platform. This device can also allow the human to imagine eyes for the Flying CoWorker and thus interpret its gaze. Thus, by orienting the camera of the Flying CoWorker towards the human we can communicate an intention to interact. The camera being fixed in relation to the base, it is thus necessary to orient the base to orient the camera and thus the gaze of the Flying CoWorker.

An autonomous aerial manipulator like the Flying CoWorker approaching a human for a handover is constrained by several aspects. The final configuration of the Flying CoWorker dictates the orientation of the arm towards the human or

the planned position for the exchange, while the Flying CoWorker base may be constrained by its embedded camera which needs to observe the scene to find its way around.

We propose in this section a way to determine the Flying CoWorker base orientation along the trajectory for the specific case of handover using an aerial manipulator like the Flying CoWorker. In the same philosophy as our work on Flying CoWorker navigation presented in the first part, we want of course to take into account the human-aware aspects but also the kinematic constraints, namely the orientation of the base θ and its angular speed $\dot{\theta}$. Concerning the human-aware aspects, we would like the Flying CoWorker to signal its presence in advance to show its intention to interact with a human for a handover.

4.4.1 Orientation calculation details

The different calculation steps are summarized in Algorithm 4. To obtain the desired behavior by simply using the orientation of the Flying CoWorker base, we propose that the Flying CoWorker starts to orient itself towards the human with whom it must interact from the moment it enters its visibility grid. As a reminder, the visibility grid considers the human's visual field but also the effort to turn the head and see the robot. Therefore, there can be an impact of the visibility grid even if the Flying CoWorker approaches behind the human's back. On the other hand, the portion of the trajectory that is not in the visibility grid, the base will tend to orient itself in the direction of its speed vector. This allows us to obtain $\overrightarrow{Direction}$, the vector pointing in the desired direction for the Flying CoWorker base. With this in mind, we also ensure the feasibility of the rotation to be performed for the base between two consecutive waypoints by calculating the desired orientation $\theta_{desired}$ that we compare with the feasible orientation from a kinematic point of view. If the desired orientation is achievable in the duration allowed to go from one waypoint to the next ($\Delta t_{segment}$), then we keep the desired value, otherwise we keep the achievable value. The duration is determined from the cartesian speed magnitude of the base and the distance traveled between the two waypoints. We perform these different steps for each waypoint of the trajectory a first time in the forward direction starting from the first waypoint. Then a second time in the backward direction from the last waypoint to ensure that we respect the orientation constraints at the start and goal.

4.4.2 Orientation behavior during approach

The obtained behavior is represented in Fig. 4.2 where the orientation of the base θ is illustrated by the blue arrows while the other colored arrows (red and green) represent its cartesian trajectory. The limit of the visibility grid is indicated by the orange dotted circle. The human is depicted in green reaching forward with his/her right hand. The starting orientation was deliberately chosen in an opposite direction in order to show the evolution of the orientation during the first moments.

Algorithm 4: Constrained orientation computation

Given;

- Orientation for last waypoint i-1: θ_{i-1}
- Speed vector of the base: $\vec{v}_{Base}$
- Flight duration of the base between waypoints i-1 and i: $\Delta t_{segment}$
- Maximum angular speed: $\dot{\theta}_{max}$

if *Flying CoWorker base pose is in visibility grid* **then**
  $\overrightarrow{Direction} = \overrightarrow{Human_{pose}} - \overrightarrow{Base_{pose}}$;
else
  $\overrightarrow{Direction} = \overrightarrow{v}_{Base}$;
end
Compute desired orientation $\theta_{desired}$ from $\overrightarrow{Direction}$ chosen;
Compute arc length δ_θ between θ_{i-1} and $\theta_{desired}$;
Estimate duration δt to traverse δ_θ: $\delta t = \frac{\delta_\theta}{\dot{\theta}_{max}}$;
if $\delta t \leqslant \Delta t_{segment}$ **then**
  $\theta = \theta_{desired}$
else
  $\theta = maximum\ feasible\ rotation\ during\ \Delta t_{segment}$
end

At the beginning, the base of the Flying CoWorker is not in the visibility grid of the human, so the orientation tends gradually to be oriented in the direction of the speed vector and then to remain aligned until the frontier of the visibility is reached. Once reached, the orientation starts to deviate towards the human to ensure that when the Flying CoWorker reaches the handover position, it is fully oriented towards the human. This moment when the Flying CoWorker starts to turn towards the human can be parameterized differently, for instance, one can decide that it starts to turn when the Flying CoWorker reaches a certain value of visibility cost or when it enters a smaller perimeter than the visibility grid. Thus, it is possible to modify the behavior of the Flying CoWorker in a very simple way by changing very few parameters.

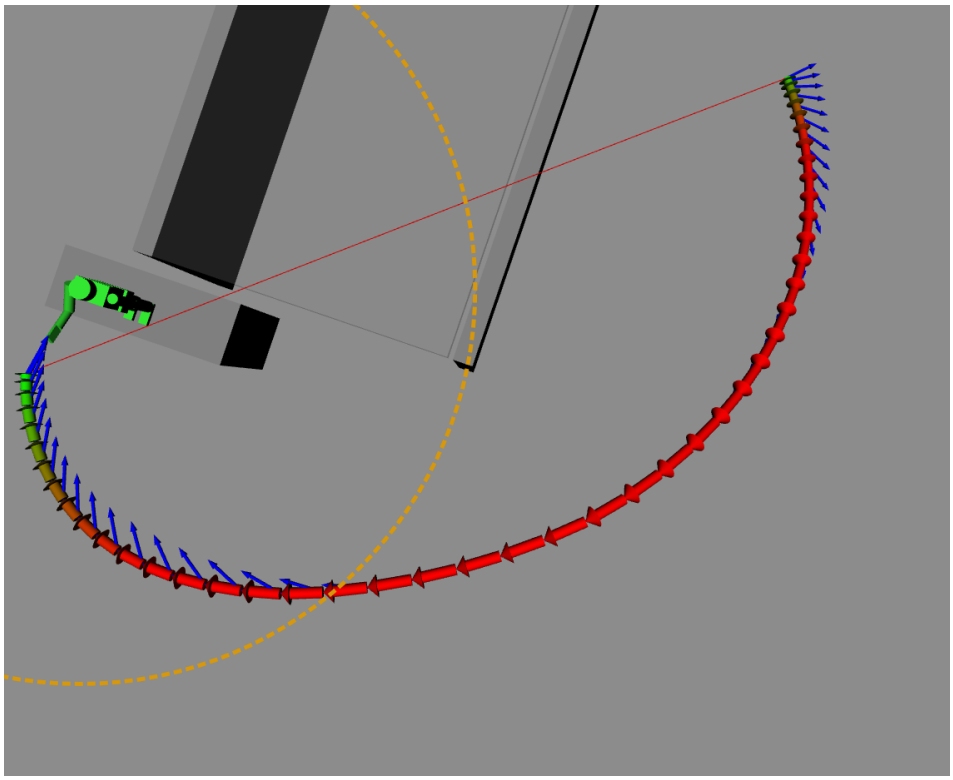


Figure 4.2: Flying CoWorker base orientation for handover illustrated by the blue arrows along the cartesian trajectory represented by red and green arrows. The base of the Flying CoWorker begins to orient itself along the trajectory before being influenced by entering the visibility grid of the human whose limit is represented by an orange dotted circle.

4.4.3 Discussion

In the case of the Flying CoWorker, the design offers a great stability of the base and thus allows us to consider only the yaw angle in our calculations. However, we can notice that Algorithm 4 works using quaternions allowing to take into account more

complex rotational motions as for acrobatic drones for example. Moreover, for the same considerations, we neglect the angular acceleration of the Flying CoWorker in our calculations. Currently, angular accelerations are taken into account using the BSpline presented in the previous chapter. To be closer to reality, future work aims at better managing the accelerations directly in our system.

4.5 Reactive FCW goal state estimation for handover

We detail in this section our proposed method to define the complete goal state of the Flying CoWorker during a handover in a reactive way. We take into consideration the human visual field and its reachability as well as safety. We also manage the static and dynamic environment as well as the dimensions of the object to be transferred.

4.5.1 Context

The FCW goal state for a handover task to be determined requires thinking about how to relate the navigation trajectory to the current situation of the human with whom the handover is to take place. This state should represent the Flying CoWorker configuration at the start of the physical interaction with the human if it is possible. Otherwise this state represents a waiting configuration of the Flying CoWorker before the human is available for handover.

How the Flying CoWorker could present itself to the human in good conditions of security and comfort during a handover. The answer to this question depends on many parameters such as:

- human's posture
- human and robot grasp positions
- geometry of the object to be exchanged
- robot's geometry and habilities
- Various obstacles in the environment

First of all, the posture of the human has a crucial impact, he/she can be standing, sitting, lying on the ground or on an object such as a scaffold. His/Her posture can also allow or not to communicate information to the robot. He/She can communicate his/her intention to the robot by holding out his/her hand, or not, or be working on an electrical panel for example, which can disturb the communication signals. The human's gaze will also be important and can be used to passively or actively guide the robot's choice. Passively, if the human looks in a direction without being aware that the robot is approaching, in this case the robot could move to the area where it will be visible. In an active way if the human looks voluntarily in a direction to propose to the robot to present itself in a place which is convenient to the human.

From the posture of the human, we can deduce and/or propose a configuration for the object so that the human can grab it comfortably without having to contort or twist his/her wrist for example. Therefore, it also impacts the grasp position of the robot itself to be able to present the object in the right configuration. This point is particularly delicate in the case of the Flying CoWorker because it is complicated to change the grasp during the flight and therefore it is possible that the grasp at the beginning of the trajectory is not ideal to respect the desired configuration for the comfort of the human at the end.

The geometry of the object to be shared will also affect the solution proposed by the planner. Let's take for example a cylindrical object, with a grasp by the robot at one end. If the object is long, the environment may constrain the possibilities and the robot may have to choose a solution that is less comfortable for the human to reach the goal, whereas a short object will offer a larger working space. Of course, the geometry of the object will also influence the grasp positions. If the object is too big for example to take it with full hand, it may be necessary to present it from above so that the human can take it from below.

The geometry of the robot will also define the solutions space. In the case of the Flying CoWorker, a rather imposing design is required to work with various objects. The base is a hexa-rotor drone that can reach 1 m in diameter, thus limiting access to certain areas or flexibility for certain configurations in cluttered environments. Habilities of the robot are directly linked to its geometry but also to the performances achievable by the hardware. The arm equipping the robot will not offer the same possibilities if it has 3 degrees of freedom as an arm with 6 degrees of freedom. The more degrees of freedom the arm has, the more complex motions it will be able to perform and therefore offer more possibilities, especially in terms of grasp positions. As well as the length of the arm which obviously limits the interactions, especially for security reasons, because if it is too short it will force the base of the Flying CoWorker to get closer, increasing the risk of collisions with the rotors.

The environment itself can have many obstacles such as walls around the human being considered for handover that can limit the solutions. The human may have to move close to these obstacles or other humans or dynamic objects may be nearby. Moreover, in the specific case of the Flying CoWorker, other parameters apply contrary to a ground robot such as the wind for example.

Moreover, all these parameters can vary in time and therefore the state of the robot planned at the beginning of the trajectory execution is likely to be different when it gets closer to the human. The need to adapt the objective frequently is therefore crucial. During the execution of the trajectory, the human can go from a situation where he/she is inaccessible because he/she is working in a cramped place for example. Then noticing that the robot is approaching to interact with him/her, he/she turns around and moves to a more accessible area to allow the exchange. The planning phase must therefore adapt in real time and not be blocked by an unexpected situation.

4.5.2 Reactive goal estimation

To answer the different points developed above, we propose a solution based on 3D cost grids and inverse kinematics. A first grid named *human_grasp_grid*, centered on the human with whom handover is considered, aims to determine the best possible handover position for the human's current posture. For each position of the *human_grasp_grid*, various configurations of the object to be exchanged are tested. From the position of the object, it is possible to determine the position of the end effector and thus generate a second grid named *base_goal_grid* in order to determine the best position for the base of the Flying CoWorker. By selecting the position of the *end_effector* and the base, it is possible to define the configuration of the Flying CoWorker's arm by inverse kinematics. Everything is updated in a very short time to allow the system to be reactive and adapt to the various changes of situation.

4.5.2.1 Design and features

human_grasp_grid Simulating the complete configuration of a robot and a human to find the best combination for a handover can be very time consuming and difficult to integrate if responsiveness is desired. On the other hand, for security and design reasons, the Flying CoWorker is very limited in the space it can move in. This is why we propose an approach where we consider only the positions reachable by the human hands for the handover. In this way, as the grid is updated with each new trajectory computed, we avoid the whole complex part aiming at estimating a good configuration of the human.

The *human_grasp_grid* shown in Fig. 4.3 thus represents potential human hand positions for the envisaged handover. Each position is associated with a cost calculated using the following cost function:

$$C_{human_grasp_grid} = \frac{C_{closest_human_hand} * (C_{vis} + C_{turn})}{C_{torso_dist}} \tag{4.1}$$

where:

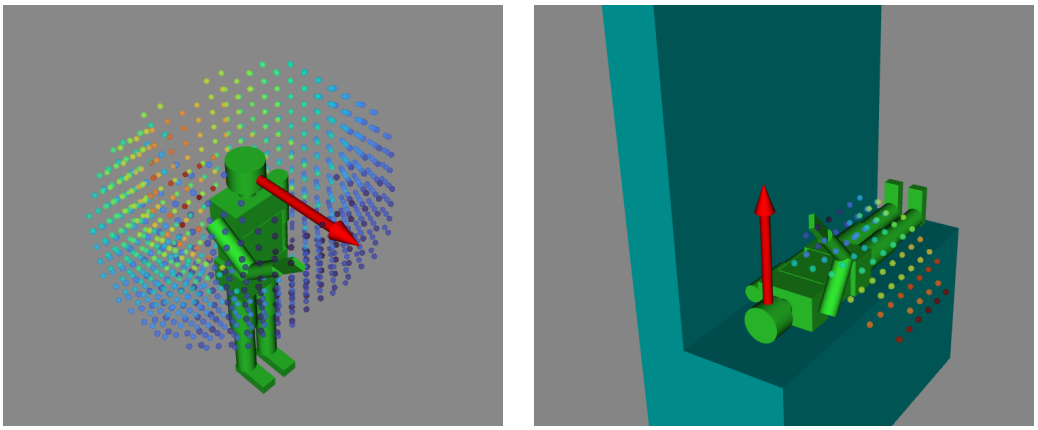
- $C_{closest_human_hand}$: distance from the cell to the nearest human hand
- C_{vis}: visibility cost defined in 2.3.2.2
- C_{turn}: angle originating in the human torso and formed between the front of the human torso and the cell
- C_{torso_dist}: distance from the cell to the human torso

By using this cost function, it is ensured that a target is found not far from one of the human's hands ($C_{closest_human_hand}$), even if it is not reachable, and that distant positions (C_{torso_dist}) are favoured for safety. Visibility and effort to see (C_{vis}) is also considered to take into account the human's gaze while a C_{turn} cost is

introduced specific to the effort required for the human to turn around regardless of the direction they are looking. In the specific case of handover where the Flying CoWorker is in the vicinity, C_{turn} allows for a greater compromise between a cell located in the human's field of view and its actual posture.

The space where costs are calculated is limited by several parameters. Firstly, there is a safety limit distance between the cell and the human head, which is considered the most sensitive part of the body. A maximum distance is determined by the distance between the cell and the shoulders of the human constrained by the length of the arms. Finally, a safety distance from obstacles is introduced to allow some freedom of flight for the system when there are walls nearby for example.

Note also that the grid is oriented to the torso and more specifically to the shoulders of the human. This set-up allows for all human postures to be considered, such as when lying down as shown in Fig. 4.3b.



(a) (b)

Figure 4.3: Representation of the *human_grasp_grid* (resolution = 0.1 meter) when the human is looking ahead. Warm colours represent high costs while cold colours represent the lowest costs a) Distribution of costs all around the human without obstacles. b) When the human is lying near obstacles the *human_grasp_grid* is dynamically adapted. Floor is represented in grey and walls in green.

shared_object Once the *human_grasp_grid* has been generated, we can start to traverse it starting from the lowest cost. This gives us a position where the human can catch the *shared_object*. In our implementation, we consider a cylindrical object that will be caught at one end by the human and at the other end by the Flying CoWorker's *end_effector*. The position corresponds to one end of the *shared_object*, and we need to define the location of the other end. To do this, we sample the surrounding space by incrementing an angle around the position to define a preferred configuration of the *shared_object*. The angle chosen here rotates around the z-axis, so we only consider horizontal positions of the cylinder.

It is possible to use an additional angle to work in spherical coordinates and allow more configuration for the object at the expense of performance. At the end of this operation, we obtain a list of configurations for the *shared_object* to which we apply the following cost function:

$$C_{shared_object} = \frac{C_{vis}}{C_{human_head_dist}} \tag{4.2}$$

where:

- C_{vis}: visibility cost defined in 2.3.2.2
- $C_{human_head_dist}$: distance between the end of the *shared_object* caught by the Flying CoWorker's *end_effector* and the human's head

This cost function thus makes it possible to favour configurations where the *shared_object* will be as most as possible in the visual field of the human while favouring distant positions for the Flying CoWorker's *end_effector*, thus providing greater safety.

base_goal_grid From the list of configurations of the *shared_object*, we now know the possible positions for the *end_effector*. Starting from the position where C_{shared_object} is minimal, we generate a new cost grid named *base_goal_grid* which this time allows us to determine the position of the Flying CoWorker base. The cost function used is similar as C_{shared_object}, simply $C_{human_head_dist}$ becomes the distance between the base of the Flying CoWorker and the human's head. As with the *shared_object*, care is taken to keep the Flying CoWorker's base as far away as possible, and therefore any rotors that might be dangerous or troublesome. For the sake of comfort, the base will be located as most as possible in areas where C_{vis} is low. Finally, we make sure that the base does not collide with the environment and humans.

As with the *human_grasp_grid*, spatial limits are imposed on the *base_goal_grid*. First of all, we ensure a minimum distance between the human's head and the base when constructing the grid. Therefore, $C_{human_head_dist}$ cannot be less than this minimum distance. The maximum distance is determined by the maximum length of the arm with which the Flying CoWorker is equipped. We also require the base to be at a higher altitude than the human head to minimise the risk of injury if the propellers are thrown off due to failure or impact.

The result can be seen in Fig. 4.4 where the position of the *human_grasp_grid* that is being considered is represented by a pink sphere close to the human's right hand. We can notice the absence of a position at the back of the human as this would force the base to be too close to the human which is not acceptable with our parameters. Another remark, the lowest costs associated to cold colours correspond well to the positions furthest from the human and in his/her visual field.

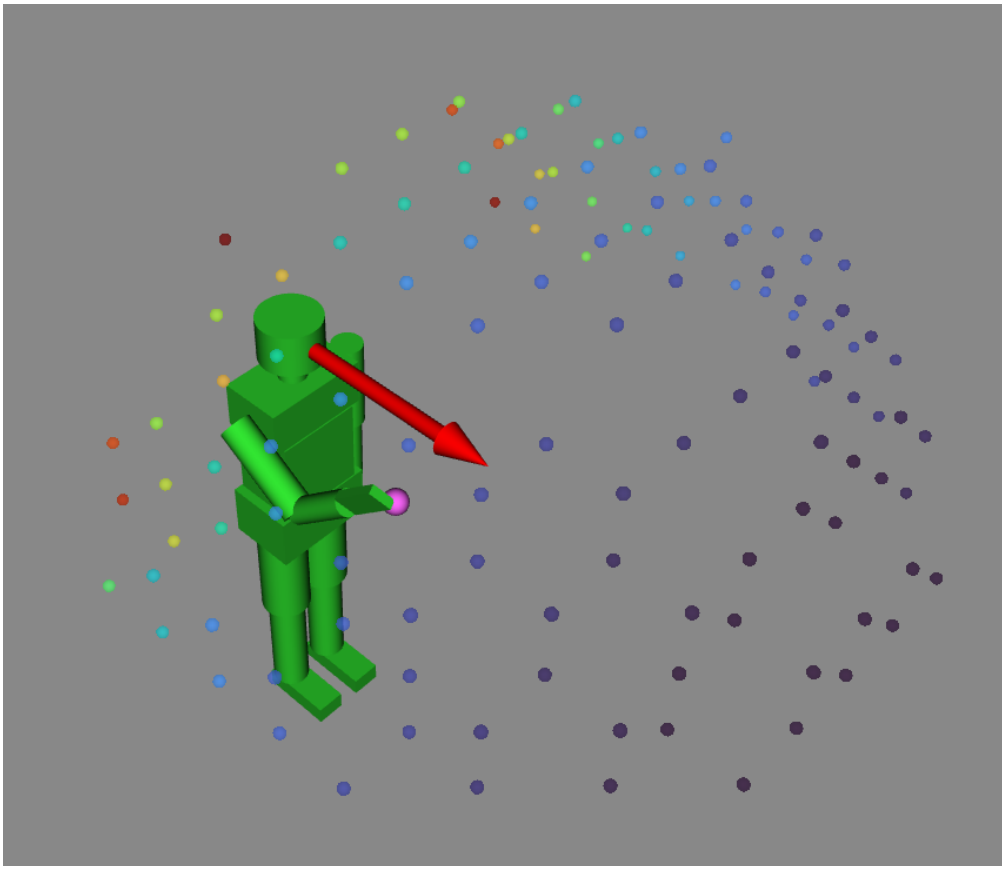


Figure 4.4: Representation of the *base_goal_grid* (grid resolution = 0.2 meter) when the human is looking ahead. The grid shows the possible positions for the base of the Flying CoWorker considering the position of the *human_grasp_grid* represented by the pink sphere near the right hand of the human. The pink sphere corresponds to the position of the *human_grasp_grid* used to generate the *base_goal_grid*. Warm colours represent high costs while cold colours represent the lowest costs. Lowest costs are better.

Flying CoWorker's arm The last step is to determine the configuration of the Flying CoWorker's arm. This is made possible by the use of inverse kinematics starting from the position of the *end_effector* and linking the position of the base. The orientation of the *end_effector* is fixed by the grasp at the *shared_object*, so the solution of the inverse kinematics is unique. We then ensure that each part of the arm respects a minimum distance to obstacles in order to provide flexibility to the system. The calculation is performed for each cell of the *base_goal_grid* to select the most interesting configuration. To do this, we sum the visibility cost ($\sum_{i=1}^{N} C_{vis}$ with N the number of links in the arm) of each part of the arm to keep the lowest cost configuration. Thus we favour the visibility of the arm by the human.

4.5.2.2 Goal examples

Let us now examine some of the results of the Flying CoWorker goal state estimation for handover generated by our implementation detailed earlier. In all the following figures, we can see a representation of the *shared_object* (in purple), the parts composing the arm (in black, red, cyan and blue) and finally the base represented by the larger blue cylinder. These cylinders represent the actual volumes considered for the collision tests, allowing us to visualize what is really taken into account by the planner rather than a mesh representation.

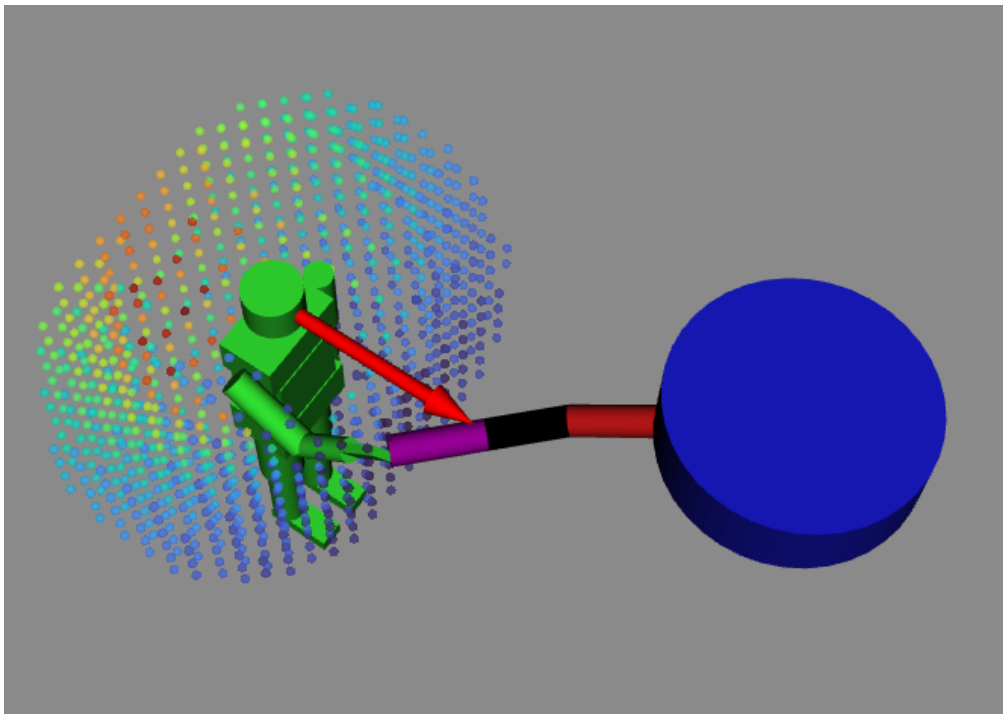
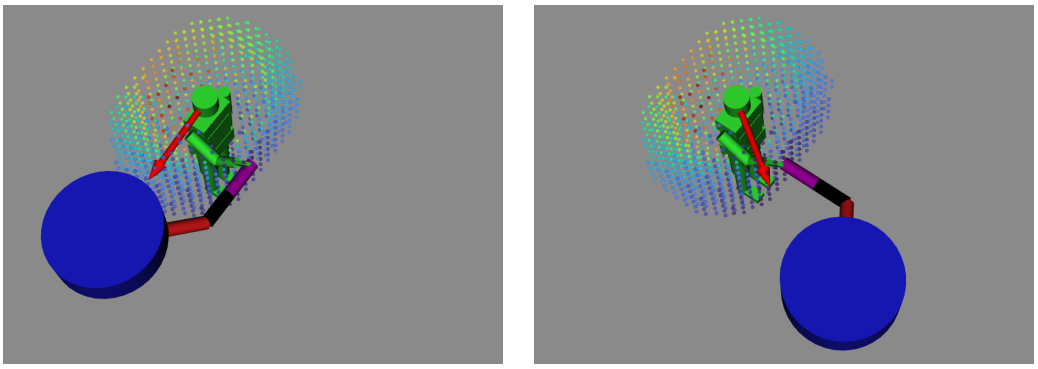


Figure 4.5: Goal configuration for a frontal handover with no obstacles and human looking ahead. Human gaze direction is represented by the red arrow. Shows the planner's ability to choose a configuration of the Flying CoWorker in the human's field of view to give the object (purple cylinder) to his/her most accessible hand. Floor is represented in grey.

Frontal handover A first situation is shown in Fig. 4.5 where the human is looking straight ahead (gaze direction is represented by the red arrow), holding out his or her right hand with no obstacles. We can see that the Flying CoWorker is located in the visual field of the human and as far away as possible considering the orientation constraint chosen for the *shared_object*. It should also be noted that the solution presented here does not offer a handover to the human's left hand. Indeed, as the human's left arm is along his/her body, his/her left hand is too close to him/her and so the planner prefers to propose a handover at the level of the

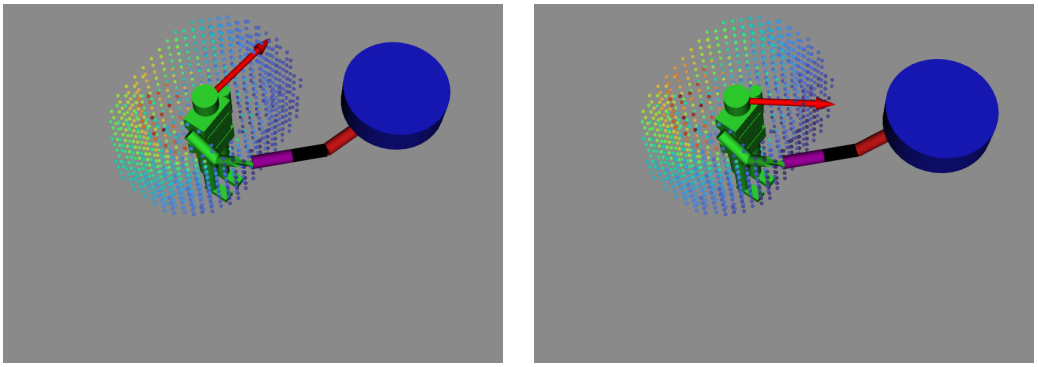
right hand, which is further away and therefore safer.

Human gaze influence We can also look at the influence of the direction of the human gaze in Fig. 4.6. In the 4 cases presented, there is no obstacle and therefore the human's hands are reachable. As a result, the end of the object to be exchanged is always at the same position, close to the human's right hand. We can therefore see that for the same solution selected in the *human_grasp_grid*, we can have many different configurations of the Flying CoWorker which will depend greatly on where the human is looking.



(a) 90° right

(b) 45° right



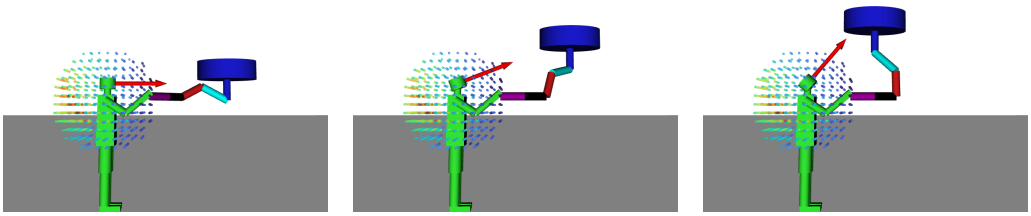
(c) 90° left

(d) 45° left

Figure 4.6: Influence of the panoramic orientation of the human head on the Flying CoWorker configuration for 4 different panoramic angle values. One end of the object is proposed towards the human's most accessible right hand position. The positions of the base and the arm of the FCW are reactively adapted to remain as much as possible in the visual field of the human. Floor is represented in grey.

The example where the human is looking 90 degrees to the right shows that the Flying CoWorker is trying to facilitate the handover by placing the object as close as possible to the human's right hand while remaining visible, especially the base which is placed in the direction of the human's field of view. The same situation arises when the human is looking about 45 degrees to the right, but this time the

base can be placed further away for more safety. In the case where the human is looking to the left, the influence of the gaze is more limited and even almost absent in the 90 degrees case. This is due to the fact that the handover position is slightly to the right of the human (right hand) and therefore the Flying CoWorker would have to get too close to the human to get directly into his/her field of view. The compromise chosen by the planner is to promote safety in this case.

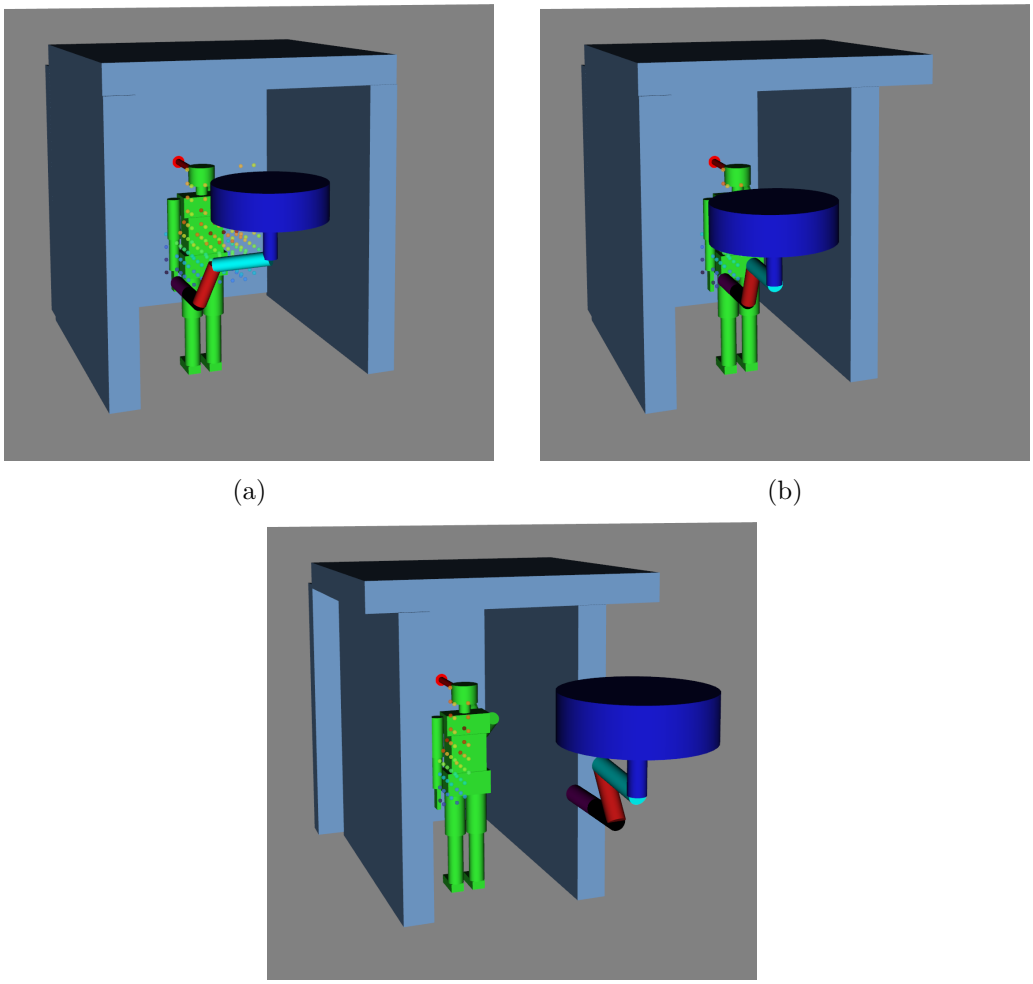


(a) 0° (b) ≈ 35° (c) ≈ 70°

Figure 4.7: Influence of the human head tilt on the Flying CoWorker configuration for 3 different tilt angle values. One end of the object is proposed towards the human's most accessible right hand position. The altitude of the FCW base is reactively adapted to remain as much as possible in the human's field of view while remaining at a distance so as not to cause discomfort to the human. Floor is represented in grey.

The upward or downward tilt of the human head influences the altitude of the Flying CoWorker base. Fig. 4.7 shows three different tilts for the same handover objective. In the case where the human is facing him/her, the base is at maximum distance and the whole robot is in his/her field of view. In the case at about 35 degrees, the base remains in his/her field of view by increasing its altitude while keeping the same position for the *end_effector*. When the human is looking up at an angle of about 70 degrees, the base further increases its altitude as far as possible because it is constrained by the arm. In addition, the planner prohibits flight positions directly over the human's head, again for safety reasons but also for comfort reasons because the wind generated by the Flying CoWorker's rotors can interfere with human vision and even project particles or dust present in the environment directly into the eyes.

Dynamic adaptation to obstacles So far, we have presented examples without barriers, leaving the planner free to choose and access both hands of the human. We therefore propose to use a more complex scenario that brings together the majority of the constraints that the Flying CoWorker might encounter when performing a handover. This scenario is the following: the human is working in front of a wall, for example on an electrical panel. The space is limited by the floor where the human is, a ceiling above his/her head but also two walls, one on his/her right and one on his/her left. This situation is illustrated in Fig. 4.8, we have kept the same configuration of the human as in the previous examples. His/Her right



(a) (b)

(c)

Figure 4.8: Behaviour of the planner when access to the Flying CoWorker is progressively reduced. a) The planner suggests placing the end of the object as close to the human's left hand as possible, as the right hand is not accessible. b) After the right wall was moved closer to the human, the Flying CoWorker configuration changed reactively to move its base away from the wall. c) After the left wall has been moved closer to the human, the space is too small to offer a direct handover solution. The planner proposes a waiting configuration away from obstacles and within the visibility grid. Floor is represented in grey and walls in light blue.

hand forward, simulating a maintenance operation on the electrical panel and not a proposed handover position as before, and his/her left hand along his/her body. As can be seen, the Flying CoWorker's access is very limited and can only approach the human from behind. From left to right Fig. 4.8, the space between the left and right walls is progressively reduced in order to observe the solutions delivered by the planner in the three cases. For all three cases, the *human_grasp_grid* is dynamically adapted to the obstacles. In the least restricted case (Fig. 4.8a), there

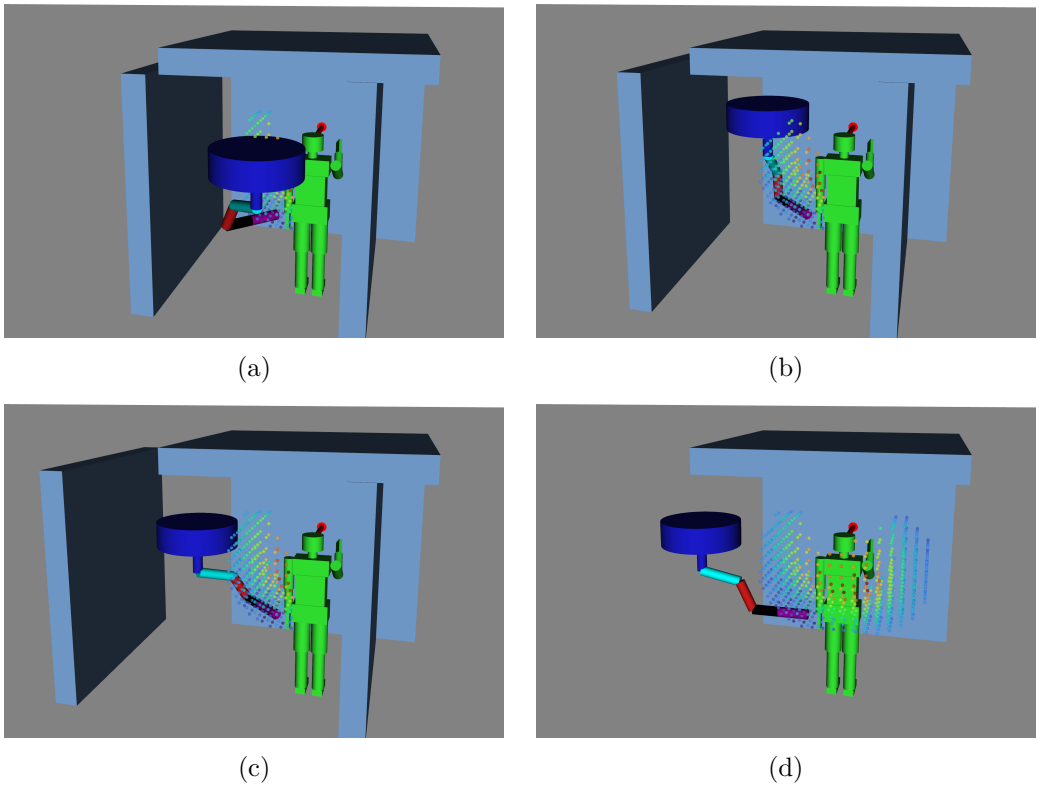


Figure 4.9: Behaviour of the planner when access to the Flying CoWorker is progressively increased. a) The space only allows a configuration in the back of the human. b) The space is large enough for the Flying CoWorker to fit on the side of the human and offer a more accessible handover. The position of the base is acceptable but close to the human. c) There is now enough room for the base to move away to cause less discomfort to the human. d) Left and right walls are removed, the chosen configuration places the base as far away as possible while offering an accessible handover position. Floor is represented in grey and walls in light blue.

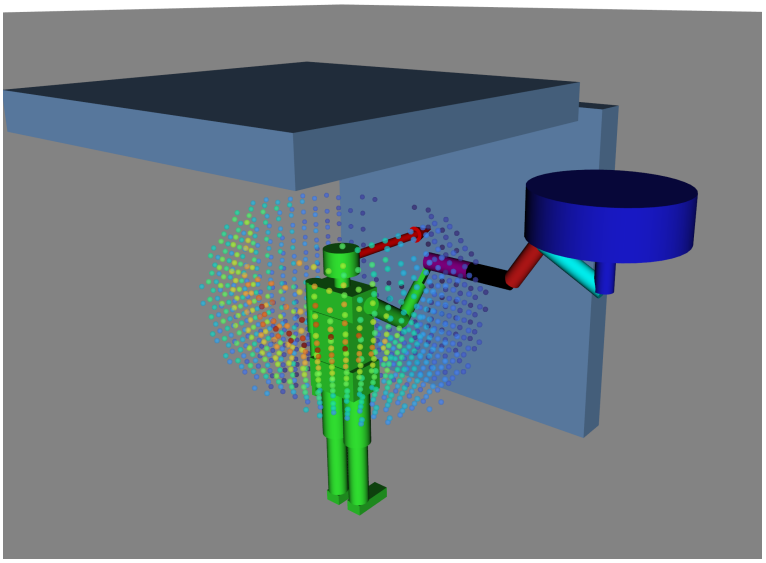
is just enough space to propose a handover close to the human's left hand knowing that the human will not see the robot if it does not turn around. In the Fig. 4.8b situation, the right wall is slightly shifted to the left reducing the space even more. It is no longer possible for the planner to propose a solution so close to the human's left hand, the position is still accessible in the *human_grasp_grid* but cannot be reached by the robot. Indeed, the base of the Flying CoWorker cannot fit between the two walls respecting the safety distances, which forces it to take a more remote position by extending the manipulator arm and by decreasing its altitude.

In the last case Fig. 4.8c, the left wall is also moved towards the human to keep the space to a minimum. We can see that there are still some positions in the *human_grasp_grid* but they are not accessible by the robot. No solution is possible for a handover in a case like this. To avoid returning an error our planner will still

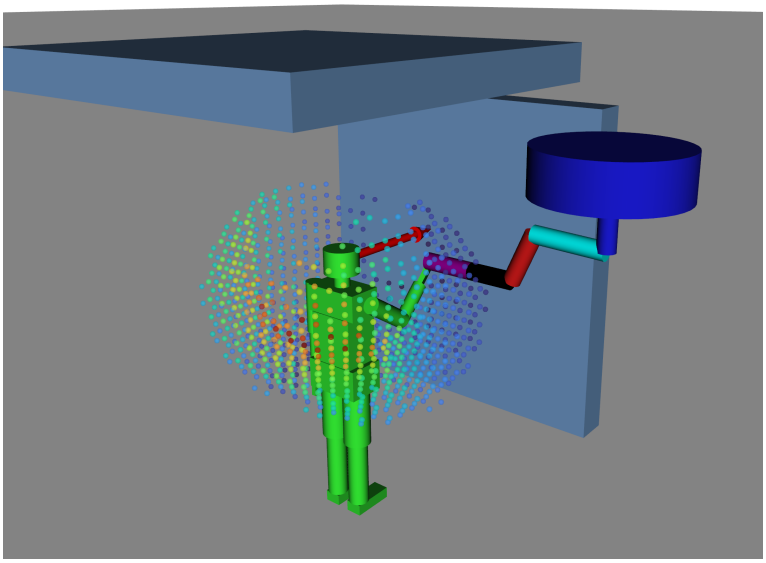
return a solution if it exists because let's imagine a situation where the human is accessible most of the time but for a moment he/she is not accessible anymore while the Flying CoWorker has already started its journey, then the trajectory must not be stopped by a temporary occultation. To meet this need, we have integrated into our planner a means of proposing a waiting position for the Flying CoWorker where it is set back from the human in the expectation that it will disengage from the situation preventing the robot from getting close enough for a handover. In our implementation, we propose the use of a fixed configuration of the arm in this waiting position, if the goal is to reduce the in-flight power consumption of the Flying CoWorker, a suitable position can be chosen. In our case, we propose that the arm is folded under the base, but another more economical configuration could be to extend the arm and the object vertically, improving the stability of the robot and thus reducing the energy required.

Once the Flying CoWorker's configuration is chosen, all that remains is to find a backward position to wait for the human's surrounding space to become available. This is achieved by using the human visibility grid defined in section 2.3.2.2. Indeed, it is preferable for the robot to wait within the range of visibility rather than behind a wall, which would prevent the human from seeing it even after turning around. The visibility grid thus allows to find an acceptable waiting place. A simple solution is to test for collisions at several locations on the grid where the visibility cost is non-zero and to keep the robot position closest to the human. To do this, we enclose the robot in a larger cylinder reducing the number of collision tests needed, improving the computation time. Thus, when the space around the human is available again, the goal will be reactively adapted and the Flying CoWorker will be able to move closer to the human for a handover.

We simulate this clearing of the space around the human in Fig. 4.9 and Fig. 4.10 by moving the walls near him/her step by step. This allows us to observe the behaviour of the Flying CoWorker and to highlight the precision and sensitivity of the planner in finding solutions according to the environment. In Fig. 4.9a, 4.9b and 4.9c, the wall to the right of the human has been moved closer to him/her in order to block the solutions in this area. On the other hand, the wall on the left is gradually shifted at each stage in order to open up the space more and more and show the solution proposed by the planner. Fig. 4.9a shows a situation where the planner proposes a handover on the human's left hand, his/her right hand being unreachable. The base of the Flying CoWorker does not have enough space to engage and place itself in the field of view. On the other hand, it is noticeable that the arm is placed in a configuration to be visible as soon as possible if the human turns around. Similarly, the object is oriented in such a way as to facilitate handover as much as possible. In Fig. 4.9b, there is just enough space for the Flying CoWorker to sneak up to the front of the human and into his/her field of vision. The operation to get to this point will be very slow because the Flying CoWorker must get between the human and the wall without risking injury. The solution is therefore feasible and acceptable with the parameters used, but the following figure (Fig. 4.9c) shows that as soon as there is enough space, the base of the Flying CoWorker will be



(a)



(b)

Figure 4.10: Behaviour of the planner when access to the human's right hand becomes available and adaptation of the Flying CoWorker's altitude. a) The right hand of the human is accessible, a configuration is proposed by the planner considering the osbtacles and the visual field of the human. b) The altitude of the ceiling above the human is increased. The proposed configuration is reactively adapted by increasing the altitude of the Flying CoWorker base to move it away from the human and cause less discomfort. Floor is represented in grey and walls in light blue.

quickly moved away to limit the risks. Note that the base of the Flying CoWorker is further away but also less in the visual field of the human than in Fig. 4.9b. Our planner thus shows a great flexibility and capacity to make compromises between human comfort and safety, which are essential when a robot must evolve close to a human and interact with him/her. The right and left walls have been removed in the last situation (Fig. 4.9d) showing that the proposed solution is still oriented towards a handover to the human's left hand. Indeed, the *human_grasp_grid* does not propose an interesting handover solution for the right hand because the wall facing the human is very close to him/her while the left hand is fully available. This behaviour is rather interesting because in this situation, we can imagine that the human has the right hand busy working while his/her left hand can recover the object that the Flying CoWorker brings him/her.

This situation allows us to introduce another example Fig. 4.10 where this time the wall facing the human is moved to clear the space and make the human's right hand accessible for a handover. We observe that the *human_grasp_grid* has been extended in front of the human and the planner proposes a handover solution on the right hand of the human with the manipulator arm extended to bring the object as close as possible and the base respecting a safe distance with the ceiling above the human's head. Looking at Fig.4.10b, we can see the reactive adaptation of the Flying CoWorker's altitude as a result of the upward displacement of the ceiling always respecting the safety distance to obstacles. Increasing the altitude of the Flying CoWorker increases the distance between the propellers and the human causing less discomfort due to the noise and wind they generate.

4.6 Coordinated motion for FCW

When a human wants to retrieve an object from a position and bring it to another human to give it to him/her, most of the time there is no discernible difference between the phase when he/she moves and when he/she hands the object to the other human. The navigational motion performed by the human's legs is coordinated with the motion of the arm(s) that allow the object to be delivered. It is a coordinated and uninterrupted motion that makes the action fluid and efficient. Wherever possible, the same should apply to a robot.

Often in robotics, handover is a stand-alone issue, with a manipulator arm whose base is fixed. In the case of a humanoid robot or mobile manipulator, it is common for the robot to perform its navigation trajectory with the arm(s) in a fixed configuration and then stop at a position close to the human before performing its handover task using a different planner, specific for planning the motion of the arm(s). In these cases the handover is therefore completely decoupled from the navigation trajectory, marking a pause in the robot's complete motion.

The Flying CoWorker is an aerial manipulator where the arm and base motions are linked. For the reasons mentioned above, we want to couple the handover phase with the navigation phase to obtain a smooth and efficient motion. For this purpose,

we propose a coordinated planning of the Flying CoWorker motions based on our KHAOS system.

In this section we propose an extension of KHAOS presented in Chapter 2 to deal with the case of coordinated motion for the Flying CoWorker performing a handover task. We first set the context before detailing the changes and improvements to the KHAOS extension. We show how the trajectories generated by KHAOS initially used for navigation are adapted to this new context of coordinated motion. We then conclude by showing some examples that reveal the capabilities of our improved planning system.

In the remainder of this chapter, when KHAOS is quoted, it refers to the modified version taking into account the coordinated motion described in this section.

4.6.1 Context

The version of KHAOS presented in Chapter 2 allows the generation of a trajectory for the Flying CoWorker navigation. This trajectory provides position and speed information in a human-aware context and respects the kinematic constraints of the Flying CoWorker. Using this first version of KHAOS, we have a way to generate a trajectory for approaching a human that will have a very smooth speed profile limiting the discomfort that will be caused to him/her.

Initially used to generate a trajectory for the Flying CoWorker base, the same algorithm can work for any object in the three-dimensional space as long as the geometry of that object is known. It is therefore also possible to generate a trajectory for the object to be delivered to the human in the handover context. The object to be delivered is the part that is closest to the human during the handover. Especially during the physical contact between the object and the human because the robot still has to hold the object so that it does not fall. It is therefore essential that the trajectory of the object to be transferred is smooth and controlled, especially in the vicinity of the human. KHAOS and its various adjustable parameters allow to obtain this result if applied to the object.

For these reasons, we propose in this section to use also a second KHAOS trajectory applied to the object to be transferred to the human. This additional trajectory provides the positions and speeds of the object from the starting point to the goal. This goal is determined using the method described in the previous section. We thus have a complete state of the Flying CoWorker at the starting point which is the current state of the robot and at the goal. We also have a trajectory for the object which serves as a reference. It remains to determine for all intermediate states, the configuration of the Flying CoWorker manipulator arm and the position of its base.

4.6.2 KHAOS main algorithm extension for coordinated motion

The overall functioning of the KHAOS algorithm is recalled in Algorithm 5. The general idea is still the same but details differ in several steps of the algorithm. We

Algorithm 5: KHAOS algorithm extension

- **Given:**
  - Start (x_0) and goal (x_N) states, $x_i \in \mathbb{R}^{DoF}$
  - An initial discretized trajectory $\boldsymbol{\theta} \in \mathbb{R}^{DoF \times N}$
  - A state-dependant cost function $q(x_i)$
- **Precompute:**
  - $\boldsymbol{A}$ = finite difference matrix
  - $\boldsymbol{R}^{-1} = (\boldsymbol{A}^T \boldsymbol{A})^{-1}$
  - $\boldsymbol{M} = \boldsymbol{R}^{-1}$, with each column scaled such that the maximum element is $(1/N)$
  - Apply Algorithm 1 to $\boldsymbol{\theta}$ to obtains associated speeds, $\boldsymbol{\theta} \in \mathbb{R}^{2 \times DoF \times N}$
- **Repeat until convergence of global trajectory cost $\boldsymbol{Q(\theta)}$:**
  1. Create K noisy trajectories, $\tilde{\boldsymbol{\theta}}_1 \cdots \tilde{\boldsymbol{\theta}}_K$ with parameters $\boldsymbol{\theta} + \boldsymbol{\varepsilon}_k$, where $\boldsymbol{\varepsilon}_k = \mathcal{N}(0, \boldsymbol{R}^{-1})$
  2. Apply Algorithm 1 to each $\tilde{\boldsymbol{\theta}}_i$ to obtains associated speeds
  3. For $k = 1 \cdots K$, compute:
     (a) Local cost $S(\tilde{\boldsymbol{\theta}}_{k,i}) = q(\tilde{\boldsymbol{\theta}}_{k,i})$ according to Equation 4.6
     (b) $P(\tilde{\boldsymbol{\theta}}_{k,i}) = \frac{e^{-\frac{1}{\lambda} S(\tilde{\boldsymbol{\theta}}_{k,i})}}{\sum_{l=1}^{K} [e^{-\frac{1}{\lambda} S(\tilde{\boldsymbol{\theta}}_{l,i})}]}$
  4. For $i \cdots (N-1)$, compute: $[\delta\tilde{\boldsymbol{\theta}}]_i = \sum_{k=1}^{K} P(\tilde{\boldsymbol{\theta}}_{k,i})[\boldsymbol{\varepsilon}_k]_i$
  5. Compute $\delta\boldsymbol{\theta} = \boldsymbol{M}\delta\tilde{\boldsymbol{\theta}}$
  6. Update $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \delta\boldsymbol{\theta}$
  7. Compute global trajectory cost $Q(\boldsymbol{\theta}) = \sum_{i=1}^{N} q(\boldsymbol{\theta}_i) + \frac{1}{2}\boldsymbol{\theta}^T \boldsymbol{R}\boldsymbol{\theta}$ according to Equation 4.7

explain these differences step by step in the rest of this section. We also identify at the beginning of each sub-section by a green writing, the line or lines of the algorithm which are concerned.

4.6.2.1 Flying CoWorker state representation

Start (x_0) and goal (x_N) states, $x_i \in \mathbb{R}^{DoF}$

The trajectory generated by KHAOS in Chapter 2 only considers the motion of the Flying CoWorker base. In this extension, we wish to generate a trajectory for the complete Flying CoWorker, i.e. the base but also the manipulator arm in addition to the object we wish to give to the human as in the proposed model presented in paragraph 4.2. For this purpose, the complete state representation of the Flying CoWorker is given as follows:

$$\text{Flying CoWorker state} \begin{cases} \left.\begin{array}{l} x_B, y_B, z_B \\ v_{x,B}, v_{y,B}, v_{z,B} \\ q_{w,B}, q_{x,B}, q_{y,B}, q_{z,B} \end{array}\right\} & \text{for the Base} \\ \mu_{i=1..6} & \text{for the arm} \\ \left.\begin{array}{l} x_O, y_O, z_O \\ v_{x,O}, v_{y,O}, v_{z,O} \\ q_{w,O}, q_{x,O}, q_{y,O}, q_{z,O} \end{array}\right\} & \text{for the Object} \end{cases} \tag{4.3}$$

which includes the positions x_B, y_B, z_B (x_O, y_O, z_O), the speeds $v_{x,B}, v_{y,B}, v_{z,B}$ $(v_{x,O}, v_{y,O}, v_{z,O})$, and then the quaternions $q_{w,B}, q_{x,B}, q_{y,B}, q_{z,B}$ $(q_{w,O}, q_{x,O}, q_{y,O}, q_{z,O})$ for the base (the object respectively). Manipulator arm joint states are also represented by μ_i. The initial input states of Algorithm 5, x_0 and x_N refer to this representation as well as all states calculated later in the algorithm.

4.6.2.2 Initial path adaptation

An initial discretized trajectory $\boldsymbol{\theta} \in \mathbb{R}^{DoF \times N}$

The state considered for the Flying CoWorker evolves in this extension as presented in 4.3. The means of generating the initial path therefore also changes. The method proposed in 2.3.1 shows how to generate an initial path. This path is composed of a list of positions for the Flying CoWorker base allowing it to connect its starting point and the arrival point by travelling a minimum distance and avoiding obstacles.

We propose in this extension to use the same method as in 2.3.1 by generating an initial path not only for the base but also for the object, both smoothed by the method presented in Chapter 3. An example of result of the generation of these two initial paths is shown in Fig. 4.11 which takes the example of the goal state presented in Fig. 4.8a. The start state x_0 and goal state x_N are also shown. The

starting state is given by the current position of the Flying CoWorker and the goal state is given by the method presented in section 4.5.

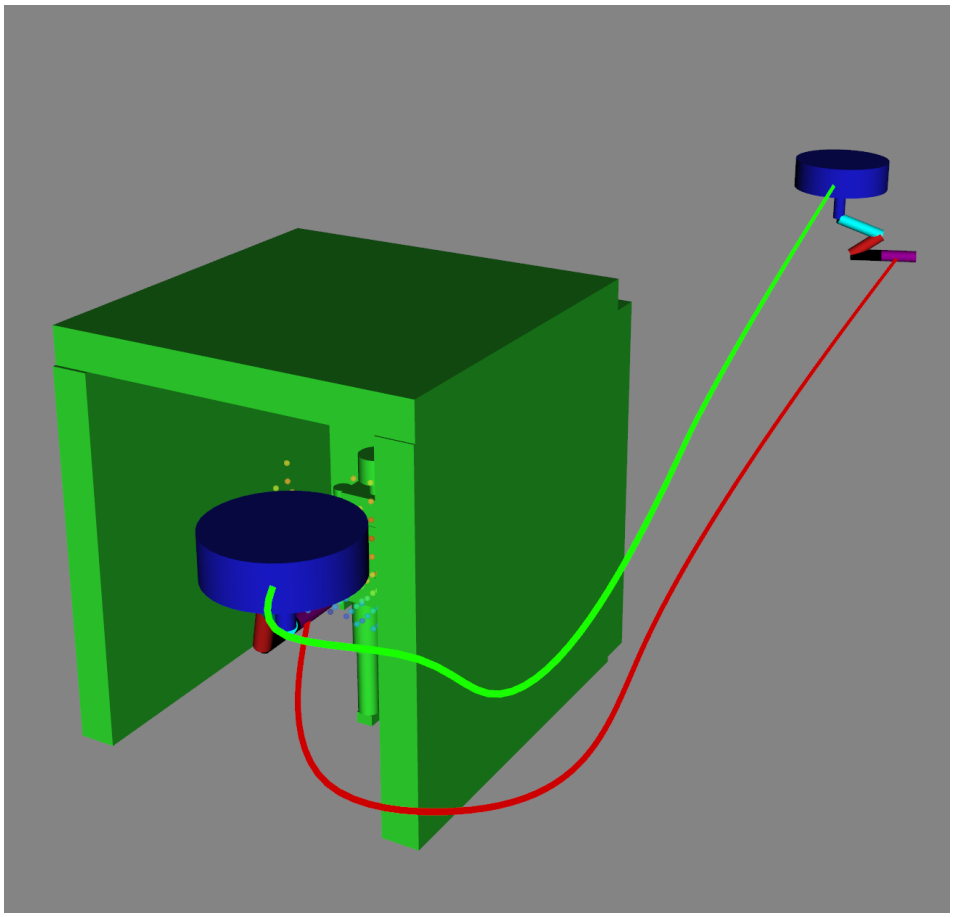


Figure 4.11: Initial path generated for the Base (green) and the object (red)
Bottom left: Flying CoWorker goal state – Top right: Flying CoWorker start state
Floor is represented in grey and walls in green.

At this point, we have two paths consisting of a list of positions for the base and the object as well as the complete start and goal state of the Flying CoWorker. To guide the optimisation algorithm, we want to convert this list of positions into a list of states of the Flying CoWorker. It is therefore necessary to determine the configuration of the arm for each pair of base and object positions. To obtain the arm configuration, we can use forward or inverse kinematics as used in the previous section where we estimate the arm configuration for the goal state. In the case of the estimation of the goal state, the position and orientation of the object is defined, which fully define the position and orientation of the end effector. As we only consider the closest solution, only one solution is defined for the arm configuration. For all intermediate positions that are not the start and goal positions, the orientation of the object is not known. These orientations must therefore be determined in order to finally calculate the joint states of the Flying CoWorker arm using inverse

Algorithm 6: Object's quaternion determination

$step_parameter = \frac{1}{N}$
for $i = 1;\ i < N;\ i++$ **do**
  $Q_{object,i} = Slerp(i * step_parameter, Q_{object,start}, Q_{object,goal})$;
end

kinematics.

To define these orientations, we propose to use the starting and goal orientations of the object and to perform a quaternion interpolation. Spherical Linear Interpolation (Slerp) is a method introduced by Shoemake [Shoemake 1985] which refers to a motion at constant speed along an arc of a circle of unit radius, given a start and end point and using an interpolation parameter between 0 and 1.

Using this method, the quaternion of the object is defined by Algorithm 6 where *step_parameter* is the interpolation step between 0 and 1. It depends on the number of states N composing the initial discretized trajectory θ given as input to algorithm 5. $Q_{object,i}$ is the quaternion of the object for state i. $Q_{object,start}$ and $Q_{object,goal}$ are the quaternions of the object for the start and goal states respectively.

4.6.2.3 Costs and constraints

A state-dependant cost function $q(x_i)$

The costs and constraints used in the initial KHAOS version from Chapter 2 are reused in its extension for coordinated motion. As a reminder:

- C_{dis} : *discomfort_cost* defined in section 2.3.2.1
- C_{vis}: *visibility_cost* defined in section 2.3.2.2
- $C_{obstacle}$: cost related to collisions with the environment
- C_{time} and C_{time_local}: time costs related to the duration of the trajectory
- C_{smooth}: related to trajectory smoothness
- Kinematic constraints defined in section 2.3.3

In addition to the costs and constraints already presented, we propose to introduce two additional costs.

Torque variation cost The Flying CoWorker base is the element that allows the motion of the whole drone. The arm must allow the position of the object to be adapted during the motion of the base so that the trajectory of the Flying CoWorker, and consequently that of the object, is smooth.

The arm is composed of several degrees of freedom. Therefore, in order to achieve a Cartesian motion of the object, several mechanical elements must be set

in motion. It is therefore easier for the Flying CoWorker base to make this motion than for the arm. For this reason, it is preferable to limit the motion of the arm along the trajectory as much as possible. To meet this need, we propose to use a torque variation cost C_{torque} defined as follows:

$$C_{torque} = \sum |\dot{\mu}_i| \tag{4.4}$$

Where $\dot{\mu}_i$ is the time variation of the joint state i of the Flying CoWorker arm. C_{torque} is the sum of the absolute values of these variations for the whole arm.

Gap cost The various costs and constraints initially introduced in KHAOS aim at deforming a trajectory. In its extension presented in this chapter, one can consider that there are two trajectories to deform, that of the base and that of the object. As a result, situations can arise where the object's trajectory tends to be too close to the base's trajectory, leaving little flexibility for the object to collide with the base.

On the other hand, it can be interesting to manage the difference in altitude between the base and the object in order to favour a configuration of the arm when it is not constrained by the configuration of the start and goal, i.e. during navigation. We propose to use a gap cost C_{gap} defined as follows:

$$C_{gap} = z_O - z_B - d \tag{4.5}$$

where z_O is the altitude of the object, z_B is the altitude of the base and d is a configurable distance allowing the gap between the object and the base to be set. The gap cost C_{gap} is positive when the object is at an altitude greater than that of the base plus the distance d. Thus, by modifying the distance d, we can for example favour a configuration of the arm extended downwards which can be interesting to improve the stability of the Flying CoWorker and reduce its energy consumption in flight.

Local cost

$$\textit{Local cost } S(\tilde{\boldsymbol{\theta}}_{k,i}) = q(\tilde{\boldsymbol{\theta}}_{k,i})$$

Equation 2.2 for calculating the local cost is modified to incorporate the two new costs presented above:

$$S(\tilde{\boldsymbol{\theta}}_{k,i}) = \begin{cases} \sum (C_{vis} + C_{dis} + C_{time_local} + C_{torque} + C_{gap}) & \text{if no collision} \\ C_{obstacle} & \text{else} \end{cases} \tag{4.6}$$

where C_{vis}, C_{dis} and C_{time_local} are the human-aware costs defined in the Chapter 2 with the difference that these costs are cumulated for the base but also for the object in this extension ($C_{vis} = C_{vis_Base} + C_{vis_Object}$ for example).

Global cost

Compute global trajectory cost $Q(\boldsymbol{\theta}) = \sum_{i=1}^{N} q(\boldsymbol{\theta}_i) + \frac{1}{2}\boldsymbol{\theta}^T \boldsymbol{R}\boldsymbol{\theta}$

As with the calculation of the local cost, equation 2.3 for calculating the overall cost of the trajectory is adapted to include the new costs:

$$Q(\boldsymbol{\theta}) = \sum_{i=1}^{N}(C_{vis} + C_{torque} + C_{gap}) + C_{time} + C_{smooth} \tag{4.7}$$

where C_{vis}, C_{time} and C_{smooth} are the costs defined in the Chapter 2. Again, these three costs are also adapted to consider the Flying CoWorker base as well as the object.

Kinematic contraints

Apply Algorithm 1 to $\boldsymbol{\theta}$ *to obtains associated speeds,* $\boldsymbol{\theta} \in \mathbb{R}^{2\times DoF\times N}$
Apply Algorithm 1 to each $\tilde{\boldsymbol{\theta}}_i$ *to obtains associated speeds*

The trajectories generated by KHAOS are constrained by Algorithm 3 detailed in Chapter 2. Initially planned for the Flying CoWorker base, we propose in this extension to apply these same constraints to the object. This makes it possible to control the smoothness of the motion of the object as well as that of the base. By controlling the smoothness of the motion of both the base and the object, we hope to achieve a smooth motion for the entire Flying CoWorker.

4.6.2.4 Noisy trajectories generation for coordinated motion

Create K *noisy trajectories,* $\tilde{\boldsymbol{\theta}}_1 \cdots \tilde{\boldsymbol{\theta}}_K$ *with parameters* $\boldsymbol{\theta} + \boldsymbol{\varepsilon}_k$*, where* $\boldsymbol{\varepsilon}_k = \mathcal{N}(0, \boldsymbol{R}^{-1})$

The major change in this extension of KHAOS is in the step 1 of Algorithm 5 which generates a large number of noisy trajectories from a reference trajectory. In this subsection we show how we determine the complete Flying CoWorker state of noisy trajectories by referring to the Cartesian trajectory of the object.

In the initial version of KHAOS, only the values of the x_B, y_B and z_B positions of the Flying CoWorker base are randomly drawn according to a normal distribution to noisy the reference trajectory. As explained earlier, in this approach we propose to use the object trajectory as a "guide" trajectory. In this way, we ensure that the object will follow a Cartesian trajectory allowing it to approach the human without any abrupt motion for handover.

Noisy trajectories are generated as shown in Fig. 4.12. For readability, we show only the state corresponding to a single waypoint of the trajectory. On the left is the reference trajectory θ composed of the complete Flying CoWorker state for the considered waypoint. As a reminder, this reference trajectory θ is determined as detailed in subsection 4.6.2.2. In the middle, the K noisy states for the considered waypoint are represented. Note that only the position of the object ($\tilde{x}_O$,$\tilde{y}_O$,$\tilde{z}_O$) and

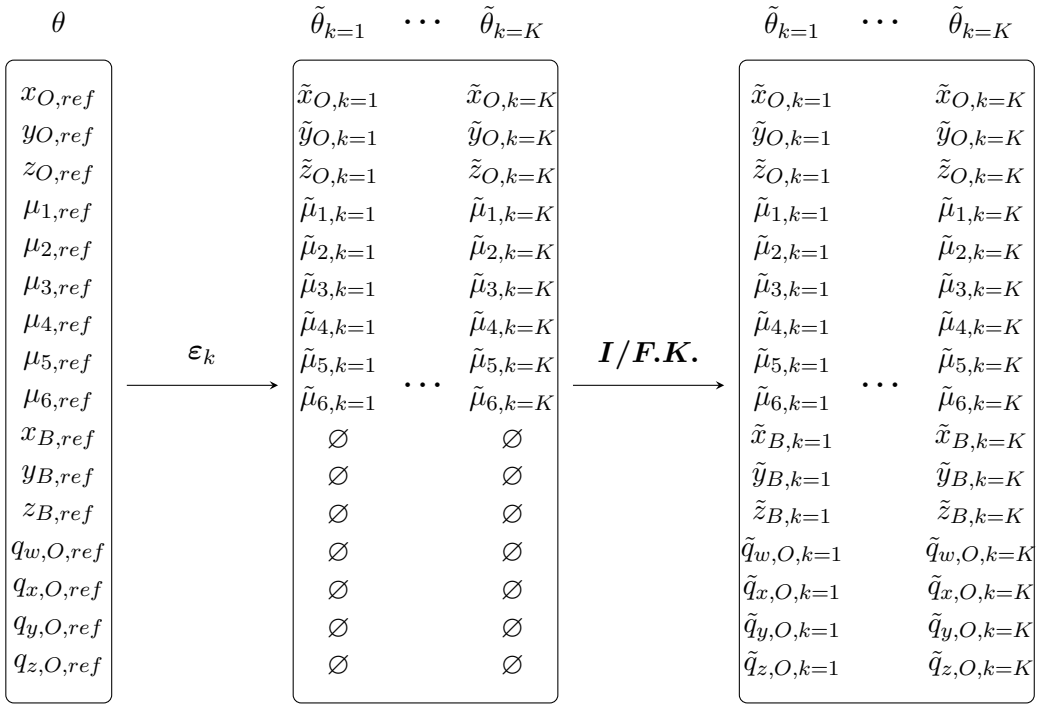


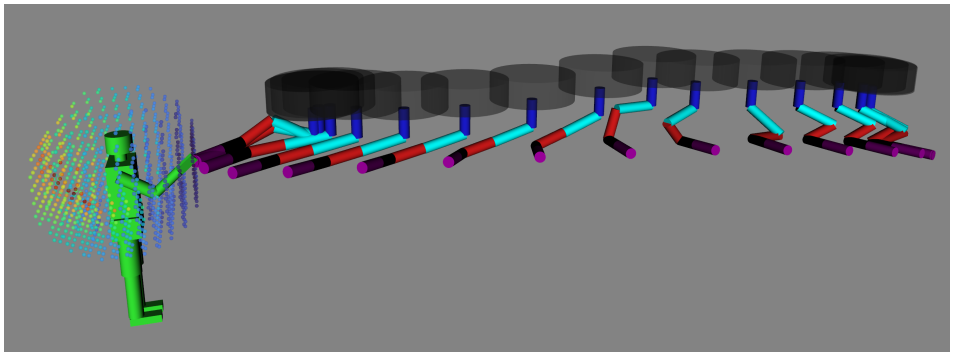
Figure 4.12: Noisy trajectories generation with complete state of the Flying CoWorker using Inverse and Forward Kinematics. ∅ corresponds to a state not yet defined.

the different joint states of the FCW arm ($\tilde{\mu}_{1..6}$) are noisy. Starting from the noisy joint states $\tilde{\mu}_{1..6}$ of the arm, we can determine the position of the base ($\tilde{x}_B$,$\tilde{y}_B$,$\tilde{z}_B$) using forward kinematics. Proceeding in this way, we ensure that we find a correct position for the base according to the object position since it is determined for a given arm configuration. Knowing the arm configuration, we can also deduce the orientation of the object ($\tilde{q}_{w,O}$, $\tilde{q}_{x,O}$, $\tilde{q}_{y,O}$ and $\tilde{q}_{z,O}$). On the right are the K full noisy states of the FCW with the position of the base and the orientation of the object.

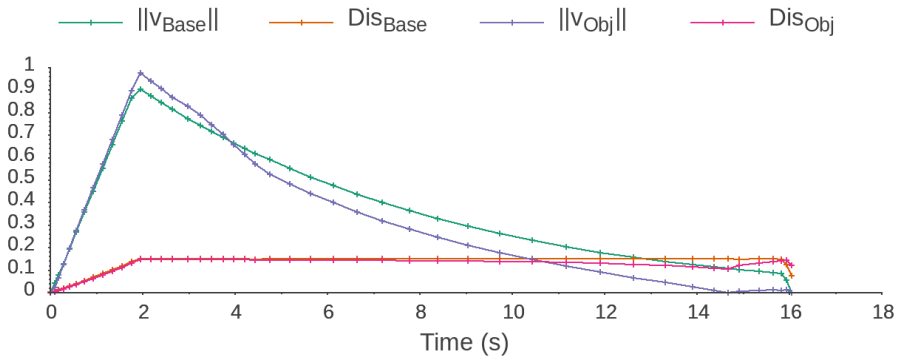
Once these noisy states are obtained, the speeds of the object and the base as well as the orientation of the base are determined by step 2 of the Algorithm 5. After this, we obtain a large number of trajectory samples to start the estimation phase of the previously defined costs. Then we proceed with the optimisation process to provide a human-aware trajectory that respects the kinematic constraints of the Flying CoWorker.

4.6.3 Coordinated handover results

After having detailed the functioning of our extension to KHAOS concerning the coordinated motion for the execution of a handover task, we present below some results showing the capabilities of our planner. In the various figures presented in



(a)



(b)

Figure 4.13: Frontal approach to the human by the Flying CoWorker to perform a handover task. a) Illustration of the coordinated motion. b) Curves of the speed magnitudes of the FCW base and the object with the corresponding *discomfort_cost*. The Flying CoWorker speed decreases very early and smoothly, limited by the discomfort constraint set to 0.15. The motion is coordinated and never interrupted. Floor is represented in grey.

this section, the base of the Flying CoWorker is shown in transparency to facilitate the readability of the robot motions. The other links composing the Flying CoWorker and the object are represented as the proposed model described in section 4.2. Again for reasons of readability, only one third of the Flying CoWorker states are shown in the figures. Moreover, in the different examples, the length of the object (purple cylinder) is 0.3 m. The last example presented in this section shows the consideration of an object of 1 m long.

Frontal approach A first example concerns a frontal approach to the human by the FCW for a handover task as illustrated in Fig. 4.13. In this context, the FCW moves into an area of the human's visual field where the *visibility_cost* varies slightly and therefore does not impact the shape of the trajectory.

The coordinated motion of the Flying CoWorker is shown in Fig. 4.13a. The starting state of the Flying CoWorker is deliberately chosen in such a way as to

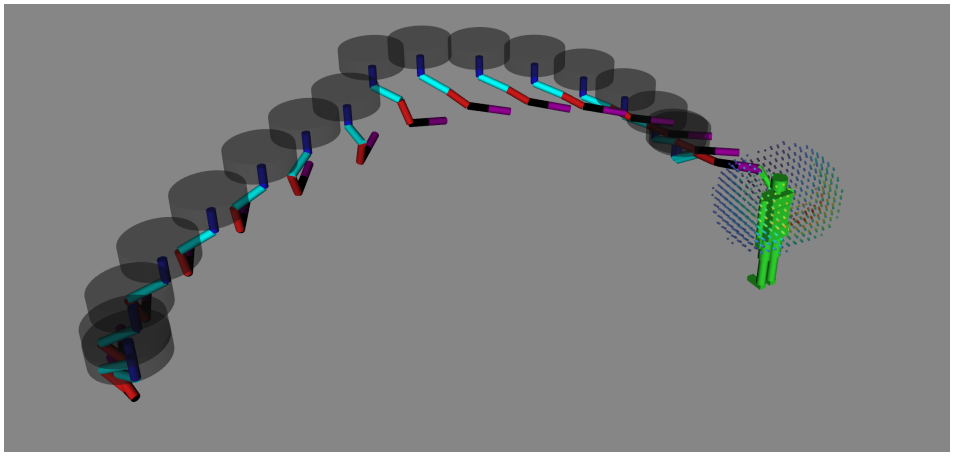
force the planner to generate a solution where the Flying CoWorker has to turn around before being in proximity to the human. In this way we can challenge our system and show its ability to simultaneously adapt the positions of the base and the object to perform a smooth and readable motion for the human. The goal state is determined by the method described in section 4.5. We can observe that the object is gradually rotated towards the goal. Thus we avoid the Flying CoWorker to stop near the human to adjust the position of the object. In the same way, we reduce the execution time of the trajectory by avoiding this stopping time. Moreover, this stopping time could be annoying for the human who may wonder what the robot's intention is.

The magnitudes of the Flying CoWorker base and object speeds and the corresponding *discomfort_cost* are shown in Fig. 4.13b. These curves allow us first of all to observe that the *discomfort_constraint* fixed here at 0.15 is never violated. We recall that the *discomfort_constraint* is used to limit the speed of the robot according to the estimated discomfort generated to the human. We can see here that the speed is limited very early by the *discomfort_constraint*, only two seconds after the Flying CoWorker departure. The deceleration is very gradual with a higher deceleration for the object than for the base. This is explained by the fact that the object has to be presented to the human and therefore is supposed to be closer to the human than the base.

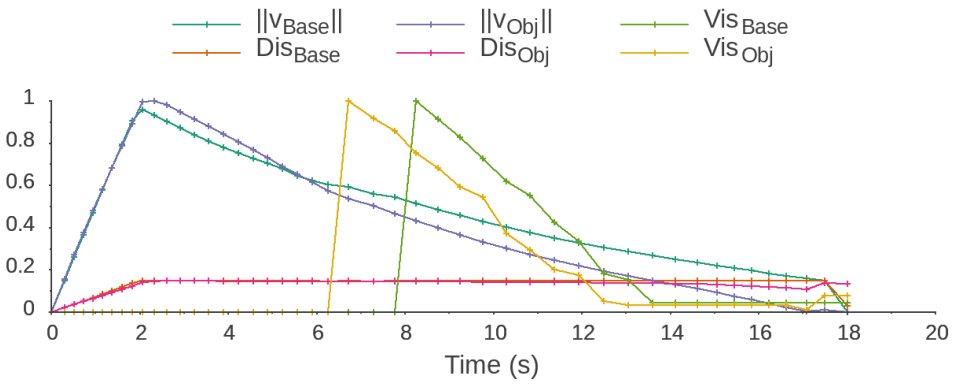
At around 15 seconds, we notice that the speed of the object is very close to zero. This is due to the fact that the object is almost in its goal state while the base must slowly move closer to meet the *discomfort_constraint*. This phase corresponds to the moment towards the end when the Flying CoWorker arm folds back to allow the base to move into the desired position. The position of the base is determined by the method in section 4.2 which requires the base and arm to be as visible as possible to the human.

Approach from the human side In this second example, we present Fig. 4.14 a situation where the Flying CoWorker starts in a state that is not directly in the human's field of view. It starts from the left of the human for an estimated goal in front of him/her as shown in Fig. 4.14a. We observe the deformation of the trajectory of the Flying CoWorker which shifts to place itself as soon as possible in the visual field of the human. A natural human reaction would be to turn his/her head to the left when he/she hears noise, mainly from the Flying CoWorker rotors. As he/she turned his/her head he/she would see the FCW appear. By shifting early to the area in front of the human, the Flying CoWorker prevents the human from turning its head too far. This behaviour reduces the discomfort caused to the human and avoids the surprise effect.

The Flying CoWorker bypasses the human to finish its trajectory facing him/her, taking care to place its base as far away as possible from him/her and being as visible as possible. In this configuration, the arm is extended to allow a secure object handover area. This keeps the base of the Flying CoWorker as far away as



(a)



(b)

Figure 4.14: Flying CoWorker approaches the human from his/her left for a handover. a) Illustration of the coordinated motion. b) Curves of the speed magnitudes of FCW base and the object with the corresponding *discomfort_cost* and normalized *visibility_cost*. The Flying CoWorker's trajectory is deviated so that it appears very early in the human's visual field in order to signal its presence and its intention to interact with him/her. Floor is represented in grey.

possible, preventing the wind generated by the rotors from causing discomfort to the human during the exchange. With the rotors away from the human hands, the risk of injury in the event of failure or breakage is reduced.

Here again we notice that the object is oriented very early on towards the human, ready for the exchange. This also allows the human to understand the Flying CoWorker's intention while he/she is several meters away.

This situation allows us to show that the trajectory behaviours presented in Chapter 2 for Flying CoWorker navigation are still present in this extension. More precisely, we show that the influence of the visual field is still correctly taken into account using the *visibility_cost*.

The magnitudes of the Flying CoWorker base and object speeds and the cor-

responding *discomfort_cost* are shown in Fig. 4.14b. The *discomfort_constraint* is set at 0.15 and reached only 2 s after the start of the Flying CoWorker limiting its speed which slowly decreases. Around 7 s, the Flying CoWorker enters the visibility grid of the human. The *visibility_cost* decreases rapidly to reach its minimum at 4 s before the end of the Flying CoWorker motion. This shows that the human is warned very early of the Flying CoWorker's intention to interact.

Approach and proposal for handover in a confined space We now present a situation where the human is working on an electrical panel. Fig. 4.15 shows that he/she is in a confined space where the Flying CoWorker cannot fully engage for a handover. Furthermore, the human is facing the wall and does not offer a direct solution for a handover without first turning around.

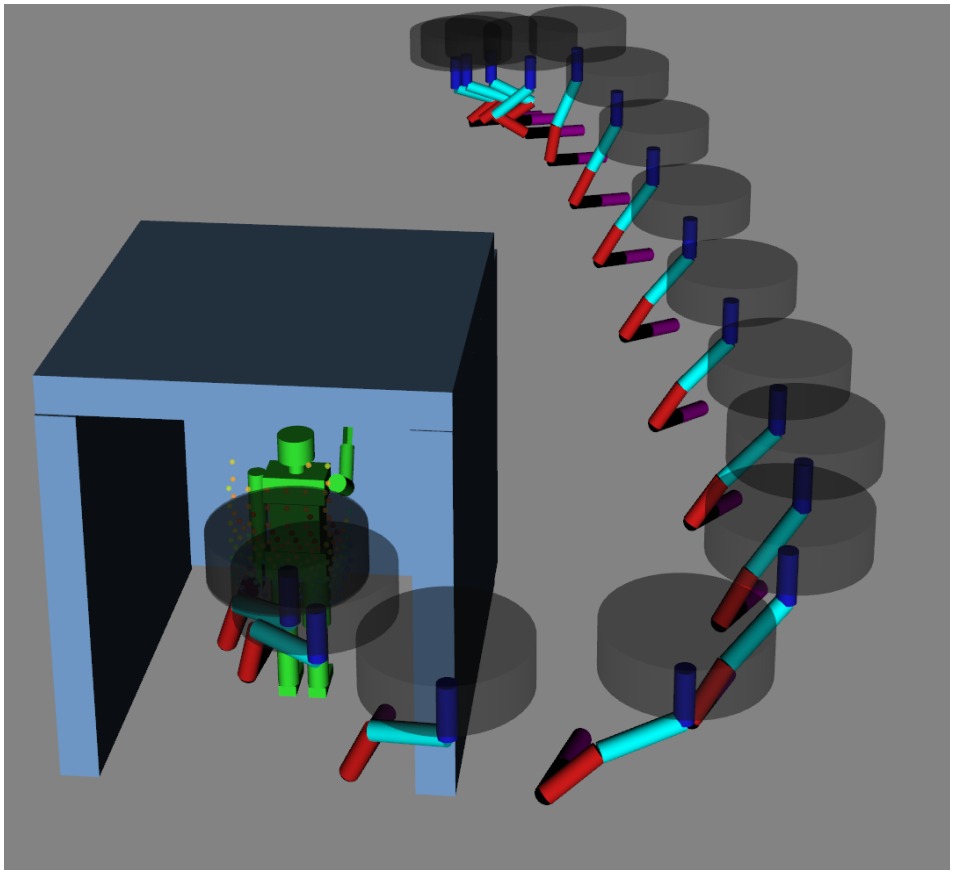
The Flying CoWorker starts from a position far behind the walls where the human is located. It must therefore bypass the obstacles and propose a handover solution. The only possibility for the Flying CoWorker is to arrive and propose a handover behind the human's back as long as he/she does not turn around.

The proposed solution is to bring the object closer to the human's left hand. It is indeed the left hand that is most easily accessible in this situation because the human's right hand is already busy working on the electrical panel. There is therefore not enough space for the planner to propose a solution in the area in front of the human. The left hand is not directly accessible either, as it is located along the human's body. KHAOS therefore proposes a solution that is slightly set back in order to maintain a safe distance.

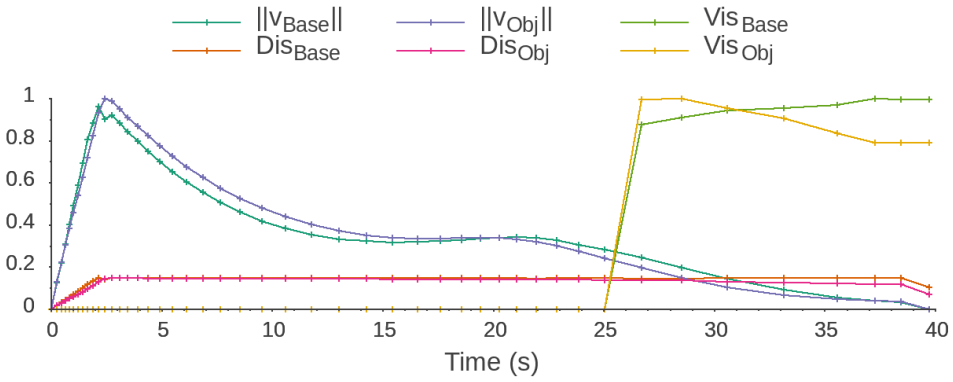
We see that the Flying CoWorker performs a coordinated motion along its trajectory. In particular, the orientation of the object is anticipated well in advance to ensure that it is presented in the correct configuration when it arrives. In effect, the Flying CoWorker must bypass the obstacle to present the object behind the human's back. It is therefore more difficult for this example to turn the object in the direction of travel and then turn it again in the direction of the human. This is because the object's initial state is already oriented towards the goal.

We can see on Fig. 4.15b that the Flying CoWorker enters the human visibility grid at around 25 s. From this moment the *visibility_cost* is and remains maximum because the Flying CoWorker is located in the back of the human. In this situation we accept that the *visibility_cost* is high to propose a handover solution. When the human will turn around to take the object, the *visibility_cost* will reduce.

The orientation of the object along the trajectory in this example is questionable. If the Flying CoWorker were to encounter other humans along its path, perhaps orienting the arm and thus the object in the direction of travel would make the trajectory more legible. In our implementation, only the orientation of the base, not shown in the figure, is oriented in the direction of travel as long as the base does not reach the visibility grid (see section 4.4 for more details). Assuming that a camera is installed on the base, it would indicate the orientation of the base and therefore the direction of travel.

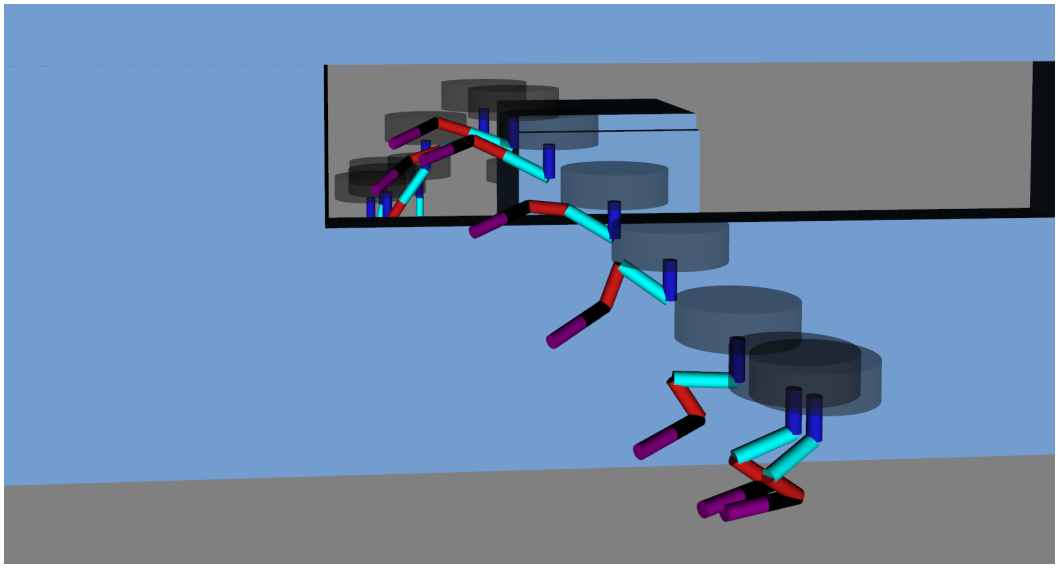


(a)

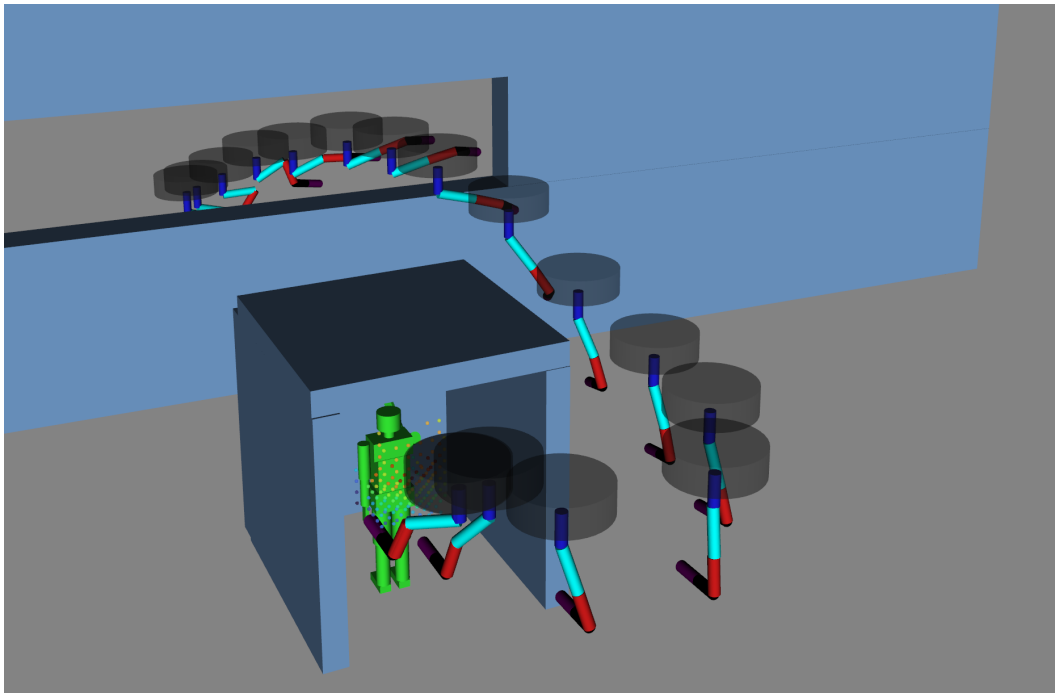


(b)

Figure 4.15: Human working on an electrical panel in a confined space. a) Illustration of the coordinated motion. b) Curves of the speed magnitudes of Flying CoWorker base and the object with the corresponding *discomfort_cost* and normalized *visibility_cost*. The Flying CoWorker's motion is coordinated and bypasses obstacles to provide a handover solution behind the human's back. Floor is represented in grey and walls in light blue.



(a) View from the start side



(b) View from the goal side

Figure 4.16: The Flying CoWorker crosses a narrow window from two different perspectives. The Flying CoWorker configuration is dynamically adapted to handle narrow passages. Floor is represented in grey and walls in light blue.

Dynamic FCW configuration through a narrow window Taking the previous situation, we increase the difficulty in this example by forcing the Flying CoWorker to navigate through a narrow passage. To do this we add two walls that

cut the Flying CoWorker's trajectory and leave only a narrow passage of one meter high as shown in Fig. 4.16 in two perspectives.

The starting and goal states of the Flying CoWorker are the same as in the previous example. Only the narrow passage therefore affects the trajectory result generated by KHAOS compared to the previous example.

We observe the coordination of the motion of the complete Flying CoWorker which folds its arm to avoid any collision as well as the base which respects a certain safety distance with the walls. Once the passage is crossed, the arm is extended again to prepare for the handover.

This example shows that our system can dynamically modify the Flying CoWorker configuration to navigate in many situations where a rigid robot could not pass. The Flying CoWorker, although large in size, can address problems similar to those presented in the work of Falanga et al [Falanga 2018]. In their work, they present a morphing system for quadrotors that consists of four arms that can fold around the main body. In our case, it is the manipulator arm that morphs the Flying CoWorker.

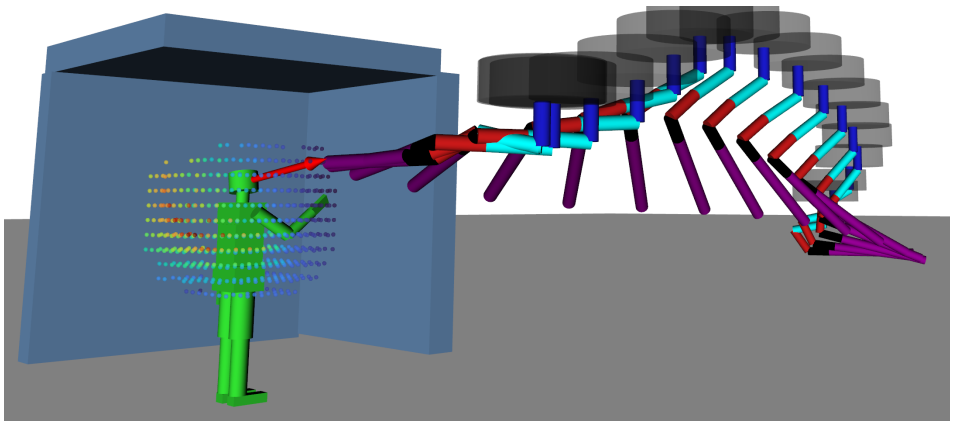


Figure 4.17: Coordinated motion of the Flying CoWorker for handover of a long object. Floor is represented in grey and walls in light blue.

Long object In this last example, we want to show the ability of KHAOS to adapt to the length of objects even if they are relatively long. We show Fig. 4.17 the coordinated motion of the Flying CoWorker for an object whose length is 1 m, which is approximately 3 times longer than in the previous examples. Goal state estimation of the Flying CoWorker is in agreement with the previous results with a smaller object. The object and the Flying CoWorker are as much as possible in the visual field of the human, the base is as far away as possible while remaining visible. Finally the end of the object is proposed close to the right hand which is accessible in this example. Despite the length of the object, KHAOS offers a smooth motion that avoids obstacles. We can notice that the Flying CoWorker takes altitude to be able to direct the object downwards before reorienting the object for handover.

Performances The trajectories generated by KHAOS in the previous examples are refreshed at a frequency around 3 Hz on a standard computer (Intel i7 1.9 GHz CPU, 32 GB memory). The integration and planning of the Flying CoWorker arm and object motions only slightly degrades the performance of KHAOS compared to its initial version (see section 2.5). The duration of the Flying CoWorker goal state estimation phase can vary between 10 Hz and 100 Hz depending on the surrounding environment of the human. Such performances allow to address in real time the complex motion planning topics presented in this manuscript. However, these performances are indicative and depend on many parameters. The convergence criterion can be adapted to obtain a more regular result by fixing the duration of the optimization phase as detailed in section 2.3.6. The factor that impacts the performance the most is the numerous collision tests performed during the goal state estimation and trajectory optimization phase. A better collision management can greatly improve the performance.

4.7 Conclusion

In this chapter we have proposed a model of the Flying CoWorker with a six degree of freedom arm. This model is defined with the aim of enriching the possibilities of the Flying CoWorker to perform a handover task.

We proposed a method to determine the orientation of the Flying CoWorker base in a handover context. This method considers the visual field of the human and the kinematic constraints of the Flying CoWorker to evaluate when the base should start orienting itself towards the human to signal its intention.

We presented a method to define the complete goal state of the Flying CoWorker for a handover task. This method studies the near-human environment to define a feasible Flying CoWorker state for a handover. It also considers various parameters already used in previous chapters such as the visibility of the Flying CoWorker by the human and the proximity. It thus allows to determine a state of the Flying CoWorker that is both feasible and comfortable for the human. This method is reactive and updates the final Flying CoWorker state in real time. It offers a solution even in cases where the human does not offer an immediate position for the exchange, such as when he/she is turned around and working on an electrical panel. This method therefore addresses situations where the receiver proposes a solution and guides the giver to the handover, but also situations where the giver has to propose a solution.

Based on the previously mentioned goal determination method, we propose an extension of our planning system KHAOS presented in chapter 2. This extension generates a human-aware trajectory and respects the kinematic constraints of the Flying CoWorker for the base but also for the arm and considers the object to be transferred. The trajectory of the object is thus controlled and very smooth allowing a secure handover and causing the least possible discomfort to the human. This extension of KHAOS also proposes a coordinated motion of the Flying

CoWorker allowing a smooth and uninterrupted motion of the whole robot. The Flying CoWorker behaviour is therefore readable by the human as soon as possible and causes little discomfort. We have presented simulation results of the coordinated Flying CoWorker motion in several situations. We have shown that the original KHAOS criteria are met for the whole robot. Finally, we presented a situation showing the ability of our system to dynamically adapt the Flying CoWorker configuration to navigate in narrow or cluttered environments.

Conclusion

The main contributions presented in this thesis are summarised below. We then discuss the perspectives for improvement and extension of the developed Kinematic Human Aware Optimization-based System for reactive planning of flying-coworker.

Contributions

We started this thesis manuscript in Chapter 1 by presenting the context of this thesis work. We detailed the purpose of "The Flying CoWorker" project, which is to have an aerial manipulator interact with humans. We stated that the work in this thesis is concerned with the motion planning and control of the Flying CoWorker. Among the objectives of the project, we presented two types of interaction scenarios to be addressed: the transport of a long object in collaboration with a human and the transport of a tool that needs to be brought to a human. This thesis work deals with the second scenario where two distinct phases have been identified. The first concerns the navigation phase of the Flying CoWorker transporting the object in a human-populated environment. The second concerns the completion of a handover task.

In Chapter 2, we introduced KHAOS: a Kinematic Human Aware Optimization-based System for reactive planning of flying-coworker. It allows reactive planning of the navigation motion of a multi-rotor drone such as the Flying CoWorker base in a human-populated environment. It takes as input an initial path composed of a list of positions that it deforms using an optimisation algorithm. The deformation is achieved with the help of social and kinematic constraints that we have made explicit. The social constraints mainly take into account the visibility of the robot by humans and the discomfort caused to them. The system output provides a waypoint list communicating position and speed information from the Flying CoWorker base. We have shown that KHAOS allows to generate a human-aware trajectory that considers the kinematic limits of the multi-rotor drone through many situations. The influence of the different constraints and the behaviour of our system are presented through these situations.

In Chapter 3, we have addressed two important issues to improve our KHAOS planning system. The first point concerns the improvement of the initial path provided as input to our system. The second point concerns the conversion of the output of KHAOS, a list of waypoints, into a time-continuous trajectory. We have highlighted the difficulty of dealing with these problems with the trajectory generators present in the literature. We proposed a solution by using a new extension of the SoftMotion library which uses Non-Uniform Cubic B-Spline. Thus, by coupling this extension of SoftMotion to KHAOS, we are able to generate time-continuous and human-aware trajectories respecting the kinematic constraints of Flying CoWorker.

In chapter 4, we presented an extension of our KHAOS planning system to handle a handover task coupled with the navigation phase. We first detailed our approach to determine the orientation of the Flying CoWorker base along the trajectory that ends in a handover phase. We then presented a method to determine the final state of the Flying CoWorker for a handover. The method reactively considers the near-human environment as well as social constraints such as visibility and safety. Knowing the final state of the Flying CoWorker using this method, we were able to present the third contribution of this chapter which deals with the coordinated motion of the Flying CoWorker along the trajectory. This extension of KHAOS thus proposes to generate a human-aware trajectory for the complete state of the Flying CoWorker while respecting its kinematic limits. This trajectory performs a coordinated motion of the arm and the base of the Flying CoWorker while considering the trajectory of the object. The object trajectory is a Cartesian trajectory allowing a smooth and comfortable motion for the approached human during the handover phase.

Perspectives

KHAOS optimisation The optimisation phase of our KHAOS planning system is based on the stochastic optimisation algorithm STOMP. Aware of the existence of a multitude of optimisation techniques, it would be interesting to explore other ways to optimise the trajectories generated by KHAOS. As shown in Chapter 3, we use an independent SoftMotion library to convert KHAOS trajectories (list of waypoints) into a time-continuous trajectory. We are keen to develop a method that incorporates B-Spline in the optimisation phase rather than the current waypoint list. This would allow us to better control the location of the B-Spline where we evaluate the costs. Moreover, we would be sure to evaluate trajectories that are already respecting the constraints of geometric and kinematic continuity.

Coordinated motion in continuous time Work is underway to generate a continuous-time trajectory for the coordinated motion presented in section 4.6 by adapting the SoftMotion extension presented in section 3.4. Indeed, the SoftMotion library extension uses B-Spline to generate a trajectory for the Flying CoWorker base only. By extending the use of B-Spline to the arm and the base of the Flying CoWorker, we could obtain a continuous-time and feasible trajectory for the whole robot.

Flying CoWorker dynamics Focusing essentially on the Human Robot Interaction aspects coupled with the kinematics of the Flying CoWorker, it would be interesting to go further on the control aspects by integrating the dynamics of the robot. From a trajectory feasibility point of view, this would certainly have an impact on the planned motion for the arm. Indeed, considering the weights of the arm and the object, we would certainly have to review the way of carrying the object

during the navigation phase to improve stability and avoid consuming too much energy.

KHAOS initial path As described in section 2.3.1, we use a basic planner to provide our system with an initial path. This initial path avoids obstacles and returns the shortest path between the start and goal. All the work required to generate a human-aware trajectory is therefore done by our system afterwards. In order to guide our system as well as possible from the first planning, it would be interesting to integrate social constraints in the planner providing the initial path. This would have an advantage especially when the environment is complex and the human-aware path would be quite different from the shortest path.

Human simulator An important aspect of evaluating a system to interact with humans is to be able to simulate those humans. There are many models or simulators of humans in the literature, but to our knowledge, none of them correspond to our needs. Either the model is very complex and difficult to implement for simple tests, or the model is not complete enough as is the case for many pedestrian simulators. The latter are often interested in navigation issues only. As a result, in terms of collision, the human is often considered as a large cylinder moving in 2D. In our case, we need a simple model for navigation, but one that distinguishes the different parts of the human, such as the head and hands, for the handover phase. Moreover, a more complete human simulator where the motion of the human's arms is simulated would have allowed us to refine our results.

Bibliography

[Berenson 2007] Dmitry Berenson, Rosen Diankov, Koichi Nishiwaki, Satoshi Kagami and James Kuffner. *Grasp planning in complex scenes.* In 2007 7th IEEE-RAS International Conference on Humanoid Robots, pages 42–48. IEEE, 2007. (Cited in page 60.)

[Bernstein 1912] Pab S Bernstein. *SUR LES RECHERCHES RECENTES RELATIVES A LA MEILLEURE APPROXIMATION DES FONCTIONS CONTINUES PAR DES POLYNÔMES.* 1912. (Cited in page 48.)

[Berscheid 2021] Lars Berscheid and Torsten Kröger. *Jerk-limited real-time trajectory generation with arbitrary target states.* arXiv preprint arXiv:2105.04830, 2021. (Cited in page 40.)

[Bevins 2021] Alisha Bevins and Brittany A Duncan. *Aerial flight paths for communication: How participants perceive and intend to respond to drone movements.* In Proceedings of the 2021 ACM/IEEE International Conference on Human-Robot Interaction, pages 16–23, 2021. (Cited in page 12.)

[Boeuf 2017] Alexandre Boeuf. *Kinodynamic motion planning for quadrotor-like aerial robots.* PhD thesis, 2017. (Cited in page 40.)

[Broquere 2008] Xavier Broquere, Daniel Sidobre and Ignacio Herrera-Aguilar. *Soft motion trajectory planner for service manipulator robot.* In 2008 IEEE/RSJ International Conference on Intelligent Robots and Systems, pages 2808–2813. IEEE, 2008. (Cited in page 49.)

[Broquere 2010] Xavier Broquere, Daniel Sidobre and Khoi Nguyen. *From motion planning to trajectory control with bounded jerk for service manipulator robots.* In 2010 IEEE International Conference on Robotics and Automation, pages 4505–4510. IEEE, 2010. (Cited in page 49.)

[Butler 2001] John Travis Butler and Arvin Agah. *Psychological effects of behavior patterns of a mobile personal robot.* Autonomous Robots, vol. 10, no. 2, pages 185–202, 2001. (Cited in page 11.)

[Cakmak 2011] Maya Cakmak, Siddhartha S Srinivasa, Min Kyung Lee, Sara Kiesler and Jodi Forlizzi. *Using spatial and temporal contrast for fluent robot-human hand-overs.* In Proceedings of the 6th international conference on Human-robot interaction, pages 489–496, 2011. (Cited in page 61.)

[Campeau-Lecours 2019] Alexandre Campeau-Lecours, Hugo Lamontagne, Simon Latour, Philippe Fauteux, Véronique Maheu, François Boucher, Charles Deguire and Louis-Joseph Caron L'Ecuyer. *Kinova modular robot arms for*

service robotics applications. In Rapid Automation: Concepts, Methodologies, Tools, and Applications, pages 693–719. IGI global, 2019. (Cited in page 58.)

[Cauchard 2015] Jessica R. Cauchard, Jane L. E, Kevin Y. Zhai and James A. Landay. *Drone & me: an exploration into natural human-drone interaction.* In Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing - UbiComp '15, pages 361–365, 2015. (Cited in pages 11 and 61.)

[Chen 2019] Changan Chen, Sha Hu, Payam Nikdel, Greg Mori and Manolis Savva. *Relational graph learning for crowd navigation.* arXiv preprint arXiv:1909.13165, 2019. (Cited in page 11.)

[Chitta 2012] Sachin Chitta, Ioan Sucan and Steve Cousins. *MoveIt! [ROS Topics].* IEEE Robotics & Automation Magazine, pages 18–19, 2012. (Cited in page 23.)

[Controzzi 2018] Marco Controzzi, Harmeet Singh, Francesca Cini, Torquato Cecchini, Alan Wing and Christian Cipriani. *Humans adjust their grip force when passing an object according to the observed speed of the partner's reaching out movement.* Experimental brain research, vol. 236, no. 12, pages 3363–3377, 2018. (Cited in page 60.)

[Corsini 2022] Gianluca Corsini, Martin Jacquet, Hemjyoti Das, Amr Afifi, Daniel Sidobre and Antonio Franchi. *Nonlinear Model Predictive Control for Human-Robot Handover with Application to the Aerial Case.* In The 2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2022), 2022. (Cited in page 61.)

[Dautenhahn 2006] Kerstin Dautenhahn, Michael Walters, Sarah Woods, Kheng Lee Koay, Chrystopher L Nehaniv, A Sisbot, Rachid Alami and Thierry Siméon. *How may I serve you? A robot companion approaching a seated person in a helping context.* In Proceedings of the 1st ACM SIGCHI/SIGART conference on Human-robot interaction, pages 172–179, 2006. (Cited in pages 11 and 61.)

[De Boor 1972] Carl De Boor. *On calculating with B-splines.* Journal of Approximation theory, vol. 6, no. 1, pages 50–62, 1972. (Cited in page 49.)

[Desormeaux 2019] Kevin Desormeaux and Daniel Sidobre. *Online Trajectory Generation: Reactive Control With Return Inside an Admissible Kinematic Domain.* In 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 4381–4386. IEEE, 2019. (Cited in page 49.)

[Dragan 2013] Anca D. Dragan, Kenton C.T. Lee and Siddhartha S. Srinivasa. *Legibility and predictability of robot motion.* In 2013 8th ACM/IEEE In-

ternational Conference on Human-Robot Interaction (HRI), pages 301–308, 2013. (Cited in page 12.)

[Duncan 2013] Brittany A. Duncan and Robin R. Murphy. *Comfortable approach distance with small Unmanned Aerial Vehicles.* In 2013 IEEE RO-MAN, pages 786–792, 2013. (Cited in page 11.)

[Falanga 2018] Davide Falanga, Kevin Kleber, Stefano Mintchev, Dario Floreano and Davide Scaramuzza. *The foldable drone: A morphing quadrotor that can squeeze and fly.* IEEE Robotics and Automation Letters, vol. 4, no. 2, pages 209–216, 2018. (Cited in page 94.)

[Ferrer 2013] Gonzalo Ferrer, Anais Garrell and Alberto Sanfeliu. *Social-aware robot navigation in urban environments.* In 2013 European Conference on Mobile Robots, pages 331–336. IEEE, 2013. (Cited in page 10.)

[Fiore 2016] Michelangelo Fiore, Aurélie Clodic and Rachid Alami. *On planning and task achievement modalities for human-robot collaboration.* In Experimental robotics, pages 293–306. Springer, 2016. (Cited in page 60.)

[Garrell 2017] Anais Garrell, Luis Garza-Elizondo, Michael Villamizar, Fernando Herrero and Alberto Sanfeliu. *Aerial social force model: A new framework to accompany people using autonomous flying robots.* In 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 7011–7017. IEEE, 2017. (Cited in page 11.)

[Garrell 2019] Anaís Garrell, Carles Coll, Renato Alquézar Mancho and Alberto Sanfeliu. *Teaching a Drone to Accompany a Person from Demonstrations using Non-Linear ASFM.* Institute of Electrical and Electronics Engineers, 2019. (Cited in page 11.)

[Gharbi 2015] Mamoun Gharbi, Pierre-Vincent Paubel, Aurélie Clodic, Ophélie Carreras, Rachid Alami and Jean-Marie Cellier. *Toward a better understanding of the communication cues involved in a human-robot object transfer.* In 2015 24th IEEE international symposium on robot and human interactive communication (RO-MAN), pages 319–324. IEEE, 2015. (Cited in page 61.)

[Goldfeder 2007] Corey Goldfeder, Peter K Allen, Claire Lackner and Raphael Pelossof. *Grasp planning via decomposition trees.* In Proceedings 2007 IEEE International Conference on Robotics and Automation, pages 4679–4684. IEEE, 2007. (Cited in page 60.)

[Guldenring 2020] Ronja Guldenring, Michael Görner, Norman Hendrich, Niels Jul Jacobsen and Jianwei Zhang. *Learning local planners for human-aware navigation in indoor environments.* In 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 6053–6060. IEEE, 2020. (Cited in page 11.)

[Hall 1966] E Hall. *The hidden dimension (anchor books a doubleday anchor book)*, 1966. (Cited in page 10.)

[Hart 2021] Justin Hart, Reuth Mirsky, Xuesu Xiao and Peter Stone. *Incorporating Gaze into Social Navigation.* arXiv e-prints, pages arXiv–2107, 2021. (Cited in page 12.)

[He 2015] Wuwei He and Daniel Sidobre. *Improving human-robot object exchange by online force classification.* Journal of Human-Robot Interaction, vol. 4, no. 1, pages pp–75, 2015. (Cited in page 60.)

[Huang 2000] Qiang Huang, Kazuo Tanie and Shigeki Sugano. *Coordinated motion planning for a mobile manipulator considering stability and manipulation.* The International Journal of Robotics Research, vol. 19, no. 8, pages 732–742, 2000. (Cited in page 62.)

[Jensen 2018] Walther Jensen, Simon Hansen and Hendrik Knoche. *Knowing You, Seeing Me: Investigating User Preferences in Drone-Human Acknowledgement.* In Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems - CHI '18, pages 1–12, 2018. (Cited in pages 11 and 61.)

[Kalakrishnan 2011] Mrinal Kalakrishnan, Sachin Chitta, Evangelos Theodorou, Peter Pastor and Stefan Schaal. *STOMP: Stochastic trajectory optimization for motion planning.* pages 4569–4574. IEEE, 2011. (Cited in pages 10, 11, 12, 17, 18, and 20.)

[Khambhaita 2016] Harmish Khambhaita, Jorge Rios-Martinez and Rachid Alami. *Head-body motion coordination for human aware robot navigation.* In 9th International workshop on Human-Friendlly Robotics (HFR 2016), page 8p, 2016. (Cited in page 12.)

[Khambhaita 2017] Harmish Khambhaita and Rachid Alami. *Viewing Robot Navigation in Human Environment as a Cooperative Activity.* In International Symposium on Robotics Research (ISSR), 2017. (Cited in pages 12 and 15.)

[Koay 2007] Kheng Lee Koay, Emrah Akin Sisbot, Dag Sverre Syrdal, Mick L Walters, Kerstin Dautenhahn and Rachid Alami. *Exploratory study of a robot approaching a person in the context of handing over an object.* In AAAI spring symposium: multidisciplinary collaboration for socially assistive robotics, pages 18–24. Stanford, CA, 2007. (Cited in pages 11, 16, and 61.)

[Konstantinova 2017] Jelizaveta Konstantinova, Senka Krivic, Agostino Stilli, Justus Piater and Kaspar Althoefer. *Autonomous object handover using wrist tactile information.* In Annual Conference Towards Autonomous Robotic Systems, pages 450–463. Springer, 2017. (Cited in page 60.)

[Kruse 2013] Thibault Kruse, Amit Kumar Pandey, Rachid Alami and Alexandra Kirsch. *Human-aware robot navigation: A survey.* Robotics and Autonomous Systems, vol. 61, no. 12, pages 1726–1743, 2013. (Cited in page 10.)

[Kruse 2014] Thibault Kruse, Alexandra Kirsch, Harmish Khambhaita and Rachid Alami. *Evaluating directional cost models in navigation.* In Proceedings of the 2014 ACM/IEEE international conference on Human-robot interaction, pages 350–357, 2014. (Cited in page 12.)

[Li 2015] Zhi Li and Kris Hauser. *Predicting object transfer position and timing in human-robot handover tasks.* Science and Systems, page 38, 2015. (Cited in page 61.)

[Li 2021] Wei Li and Rong Xiong. *A hybrid visual servo control method for simultaneously controlling a nonholonomic mobile and a manipulator.* Frontiers of Information Technology & Electronic Engineering, vol. 22, no. 2, pages 141–154, 2021. (Cited in page 62.)

[Mainprice 2010] Jim Mainprice, Emrah Akin Sisbot, Thierry Siméon and Rachid Alami. *Planning safe and legible hand-over motions for human-robot interaction.* In IARP/IEEE-RAS/EURON workshop on technical challenges for dependable robots in human environments, 2010. (Cited in pages 34 and 61.)

[Mainprice 2012] Jim Mainprice, Mamoun Gharbi, Thierry Siméon and Rachid Alami. *Sharing effort in planning human-robot handover tasks.* In 2012 IEEE RO-MAN: The 21st IEEE International Symposium on Robot and Human Interactive Communication, pages 764–770. IEEE, 2012. (Cited in page 61.)

[Mavrogiannis 2021] Christoforos Mavrogiannis, Francesca Baldini, Allan Wang, Dapeng Zhao, Pete Trautman, Aaron Steinfeld and Jean Oh. *Core Challenges of Social Robot Navigation: A Survey*, 2021. (Cited in page 1.)

[May 2015] Alyxander David May, Christian Dondrup and Marc Hanheide. *Show me your moves! Conveying navigation intention of a mobile robot to humans.* In 2015 European Conference on Mobile Robots (ECMR), pages 1–6. IEEE, 2015. (Cited in page 12.)

[Medina 2016] José R Medina, Felix Duvallet, Murali Karnam and Aude Billard. *A human-inspired controller for fluid human-robot handovers.* In 2016 IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids), pages 324–331. IEEE, 2016. (Cited in page 60.)

[Miller 2003] Andrew T Miller, Steffen Knoop, Henrik I Christensen and Peter K Allen. *Automatic grasp planning using shape primitives.* In 2003 IEEE International Conference on Robotics and Automation (Cat. No. 03CH37422), volume 2, pages 1824–1829. IEEE, 2003. (Cited in page 60.)

[Moon 2014] AJung Moon, Daniel M Troniak, Brian Gleeson, Matthew KXJ Pan, Minhua Zheng, Benjamin A Blumer, Karon MacLean and Elizabeth A Croft. *Meet me where i'm gazing: how shared attention gaze affects human-robot handover timing.* In Proceedings of the 2014 ACM/IEEE international conference on Human-robot interaction, pages 334–341, 2014. (Cited in page 61.)

[Ortenzi 2021] Valerio Ortenzi, Akansel Cosgun, Tommaso Pardi, Wesley P Chan, Elizabeth Croft and Dana Kulić. *Object handovers: a review for robotics.* IEEE Transactions on Robotics, vol. 37, no. 6, pages 1855–1873, 2021. (Cited in page 60.)

[Pan 2018] Matthew KXJ Pan, Elizabeth A Croft and Günter Niemeyer. *Exploration of geometry and forces occurring within human-to-robot handovers.* In 2018 IEEE Haptics Symposium (HAPTICS), pages 327–333. IEEE, 2018. (Cited in page 60.)

[Park 2012] Chonhyon Park, Jia Pan and Dinesh Manocha. *ITOMP: Incremental trajectory optimization for real-time replanning in dynamic environments.* In Twenty-Second International Conference on Automated Planning and Scheduling, 2012. (Cited in page 21.)

[Peternel 2017] Luka Peternel, Wansoo Kim, Jan Babič and Arash Ajoudani. *Towards ergonomic control of human-robot co-manipulation and handover.* In 2017 IEEE-RAS 17th International Conference on Humanoid Robotics (Humanoids), pages 55–60. IEEE, 2017. (Cited in page 62.)

[Piegl 1996] Les Piegl and Wayne Tiller. The nurbs book. Springer Science & Business Media, 1996. (Cited in page 47.)

[Rasch 2018] Robin Rasch, Sven Wachsmuth and Matthias König. *A joint motion model for human-like robot-human handover.* In 2018 IEEE-RAS 18th International Conference on Humanoid Robots (Humanoids), pages 180–187. IEEE, 2018. (Cited in page 61.)

[Repiso 2017] Ely Repiso, Gonzalo Ferrer and Alberto Sanfeliu. *On-line adaptive side-by-side human robot companion in dynamic urban environments.* In 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 872–877. IEEE, 2017. (Cited in page 10.)

[Rios-Martinez 2015] Jorge Rios-Martinez, Anne Spalanzani and Christian Laugier. *From proxemics theory to socially-aware navigation: A survey.* International Journal of Social Robotics, vol. 7, no. 2, pages 137–153, 2015. (Cited in page 10.)

[Rousseau 2019] Gauthier Rousseau. *Optimal trajectory planning and predictive control for cinematographic flight plans with quadrotors.* PhD thesis, Université Paris-Saclay, 2019. (Cited in page 49.)

[Rubagotti 2022] Matteo Rubagotti, Inara Tusseyeva, Sara Baltabayeva, Danna Summers and Anara Sandygulova. *Perceived safety in physical human–robot interaction—A survey.* Robotics and Autonomous Systems, vol. 151, page 104047, 2022. (Cited in page 2.)

[Sandakalum 2022] Thushara Sandakalum and Marcelo H Ang Jr. *Motion planning for mobile manipulators—a systematic review.* Machines, vol. 10, no. 2, page 97, 2022. (Cited in pages 2 and 62.)

[Saut 2012] Jean-Philippe Saut and Daniel Sidobre. *Efficient models for grasp planning with a multi-fingered hand.* Robotics and Autonomous Systems, vol. 60, no. 3, pages 347–357, 2012. (Cited in page 60.)

[Shoemake 1985] Ken Shoemake. *Animating rotation with quaternion curves.* In Proceedings of the 12th annual conference on Computer graphics and interactive techniques, pages 245–254, 1985. (Cited in page 84.)

[Sidobre 2019] Daniel Sidobre and Kevin Desormeaux. *Smooth cubic polynomial trajectories for human-robot interactions.* Journal of Intelligent & Robotic Systems, pages 851–869, 2019. (Cited in pages 34, 46, and 49.)

[Singamaneni 2020] Phani-Teja Singamaneni and Rachid Alami. *HATEB-2: Reactive Planning and Decision making in Human-Robot Co-navigation.* In International Conference on Robot & Human Interactive Communication, 2020, 2020. (Cited in page 12.)

[Singamaneni 2021] Phani Teja Singamaneni, Anthony Favier and Rachid Alami. *Human-Aware Navigation Planner for Diverse Human-Robot Contexts.* In IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2021. (Cited in page 11.)

[Sisbot 2007] E.A. Sisbot, L.F. Marin-Urias, R. Alami and T. Simeon. *A Human Aware Mobile Robot Motion Planner.* IEEE Transactions on Robotics, 2007. (Cited in pages 12 and 16.)

[Sisbot 2012] Emrah Akin Sisbot and Rachid Alami. *A human-aware manipulation planner.* IEEE Transactions on Robotics, 2012. (Cited in page 34.)

[Szafir 2014] Daniel Szafir, Bilge Mutlu and Terrence Fong. *Communication of intent in assistive free flyers.* In Proceedings of the 2014 ACM/IEEE international conference on Human-robot interaction, pages 358–365, 2014. (Cited in pages 12 and 61.)

[Truc 2022] Jérôme Truc, Phani-Teja Singamaneni, Daniel Sidobre, Serena Ivaldi and Rachid Alami. *KHAOS: a Kinematic Human Aware Optimization-based System for Reactive Planning of Flying-Coworker.* In ICRA 2022, 2022. (Cited in pages 9 and 37.)

[Truong 2017] Xuan-Tung Truong and Trung Dung Ngo. *Toward socially aware robot navigation in dynamic and crowded environments: A proactive social motion model.* IEEE Transactions on Automation Science and Engineering, vol. 14, no. 4, pages 1743–1760, 2017. (Cited in page 11.)

[Vazquez-Santiago 2021] Kyshalee Vazquez-Santiago, Chun Fan Goh and Kenji Shimada. *Motion Planning for Kinematically Redundant Mobile Manipulators with Genetic Algorithm, Pose Interpolation, and Inverse Kinematics.* In 2021 IEEE 17th International Conference on Automation Science and Engineering (CASE), pages 1167–1174. IEEE, 2021. (Cited in page 62.)

[Vianello 2021] Lorenzo Vianello, Jean-Baptiste Mouret, Eloïse Dalin, Alexis Aubry and Serena Ivaldi. *Human posture prediction during physical human-robot interaction.* IEEE Robotics and Automation Letters, vol. 6, no. 3, pages 6046–6053, 2021. (Cited in page 61.)

[Xing 2021] Hongjun Xing, Ali Torabi, Liang Ding, Haibo Gao, Weihua Li and Mahdi Tavakoli. *Enhancing kinematic accuracy of redundant wheeled mobile manipulators via adaptive motion planning.* Mechatronics, vol. 79, page 102639, 2021. (Cited in page 62.)

[Yang 2020] Wei Yang, Chris Paxton, Maya Cakmak and Dieter Fox. *Human grasp classification for reactive human-to-robot handovers.* In 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 11123–11130. IEEE, 2020. (Cited in page 60.)

[Yang 2022] Wei Yang, Balakumar Sundaralingam, Chris Paxton, Iretiayo Akinola, Yu-Wei Chao, Maya Cakmak and Dieter Fox. *Model Predictive Control for Fluid Human-to-Robot Handovers.* arXiv preprint arXiv:2204.00134, 2022. (Cited in page 60.)

[Yeh 2017] Alexander Yeh, Photchara Ratsamee, Kiyoshi Kiyokawa, Yuki Uranishi, Tomohiro Mashita, Haruo Takemura, Morten Fjeld and Mohammad Obaid. *Exploring proxemics for human-drone interaction.* In Proceedings of the 5th international conference on human agent interaction, pages 81–88, 2017. (Cited in page 11.)

[Yoon 2019] Hyung-Jin Yoon, Christopher Widdowson, Thiago Marinho, Ranxiao Frances Wang and Naira Hovakimyan. *Socially Aware Path Planning for a Flying Robot in Close Proximity of Humans.* ACM Transactions on Cyber-Physical Systems, 2019. (Cited in page 12.)

[Yoshikawa 1985] Tsuneo Yoshikawa. *Manipulability of robotic mechanisms.* The international journal of Robotics Research, vol. 4, no. 2, pages 3–9, 1985. (Cited in page 62.)

[Zhang 2016] Yunong Zhang, Xiaogang Yan, Dechao Chen, Dongsheng Guo and Weibing Li. *QP-based refined manipulability-maximizing scheme for coordinated motion planning and control of physically constrained wheeled mobile redundant manipulators*. Nonlinear Dynamics, vol. 85, no. 1, pages 245–261, 2016. (Cited in page 62.)

Résumé: Le sujet des travaux de cette thèse vise à planifier les mouvements d'un manipulateur aérien (Un héxa-rotor équipé d'un bras manipulateur) aussi appelé collaborateur volant. Il est utilisé pour intéragir avec des humains notamment pour transporter un objet et le remettre à un collègue humain.

Le transport d'un objet par le collaborateur volant jusqu'à son destinataire humain pour le lui donner nécessite d'abord de naviguer dans un environnement comportant des obstacles et potentiellement d'autres humains pouvant eux aussi être en mouvement. Au delà du problème classique de la recherche et de l'optimisation d'un mouvement sans collision, cette phase de navigation doit tenir compte des contraintes et préférences humaines. Un aspect important concerne la lisibilité et l'acceptabilité de l'intention du robot afin d'être compris par les humains environnants et notamment son destinataire. De nombreux signaux peuvent être inscrits dans la forme et la dynamique du mouvement d'un robot. Un déplacement rapide du robot en direction d'un humain peut s'avérer inquiétant pour celui-ci. De même qu'un robot qui surgit de derrière un obstacle proche d'un humain peut surprendre. La phase de remise de l'objet à l'humain doit considérer tout ces paramètres en plus du déploiement du bras qui doit proposer un transfert de l'objet en toute sécurité et considérer le confort du collègue humain. La coordination du mouvement entre la base et le bras du collaborateur volant est un atout pour améliorer la fluidité du mouvement le long de sa trajectoire et signaler son intention au plus tôt pour que les humains puissent réagir en conséquence. La position des humains et des obstacles de l'environnement étant amenées à varier fréquemment, la plannification des mouvements du collaborateur volant doit être réalisée de manière réactive et adaptative.

Nous présentons KHAOS un systême de planification réactive considérant les contraintes et préférences humaines en plus des contraintes cinématiques du collaborateur volant. Basé sur un algorithme d'optimisation stochastique, il déforme un chemin à l'aide de coût sociaux et cinématiques et génère une liste de points de passages donnant les positions et vitesses pour la navigation dans un espace 3D.

Nous utilisons un générateur de trajectoire basé sur des B-Spline dans le but de former une trajectoire continue en temps passant par les positions et respectant les vitesses donnés par la liste de points de passages de KHAOS. En procédant ainsi nous fournissons une trajectoire respectant la cinématique du collaborateur volant à chaque instant et permettant l'execution directe par un contrôlleur bas niveau.

Nous étendons ensuite les capacités de KHAOS en appliquant des contraintes supplémentaires afin qu'il puisse également planifier les mouvements du bras et de l'objet en plus de la base dans le but de naviguer et réaliser une tâche de remise d'objet. Dans cette extension, les mouvements sont planifiés de manière coordonnée permettant un mouvement fluide et sans interruption durant le déroulement de la tâche.

Abstract: The subject of this thesis is to plan the motions of an aerial manipulator (a hexa-rotor equipped with a manipulator arm) also called a Flying CoWorker. It is used to interact with humans, in particular to transport an object and deliver it to a human coworker.

The transport of an object by the Flying CoWorker to its human receiver to give it to him requires first to navigate in an environment with obstacles and potentially other humans who may also be in motion.

Beyond the classical problem of finding and optimising a collision-free motion, this navigation phase must take into account human constraints and preferences. An important aspect is the legibility and acceptability of the robot's intention in order to be understood by the surrounding humans and in particular its receiver. Many signals can be embedded in the shape and dynamics of a robot's motion. A fast motion of the robot towards a human may be alarming to the human. Similarly, a robot that emerges from behind an obstacle and close to a human can be surprising.

The handover phase must consider all these parameters in addition to the deployment of the arm, which must offer a safe handover of the object and consider the comfort of the human coworker. The coordination of the motion between the base and the Flying CoWorker's arm is an asset to improve the smoothness of the motion along its trajectory and to signal its intention as soon as possible so that the humans can react accordingly. As the position of humans and obstacles in the environment are likely to vary frequently, the planning of the Flying CoWorker's motions must be done in a reactive and adaptative manner.

We present KHAOS, a reactive planning system considering human constraints and preferences in addition to the kinematic constraints of the Flying CoWorker. Based on a stochastic optimisation algorithm, it deforms a path using social and kinematic costs and generates a list of waypoints giving positions and speeds for navigation in a 3D space.

We use a B-Spline based trajectory generator to form a continuous trajectory in time through the positions and speeds given by the KHAOS waypoint list. By doing so we provide a trajectory that respects the kinematics of the Flying CoWorker at each instant and allows direct execution by a low-level controller.

We then extend the capabilities of KHAOS by applying additional constraints so that it can also plan the arm and object motions in addition to the base in order to navigate and perform an object handover task. In this extension, the motions are planned in a coordinated manner allowing smooth and uninterrupted movement during the task.

Keywords: Human Robot Interaction, Autonomous Aerial manipulation, Coordinated motion planning, Aerial robots