Gestion de robots mobiles et redondants et collaboratifs en environnement contraint et dynamique

David Busson

2016-ENAM-0041

École doctorale n° 432 : Science des Métiers de l'ingénieur

Doctorat ParisTech

T H È S E

pour obtenir le grade de docteur délivré par

l'École Nationale Supérieure d'Arts et Métiers

Spécialité " Automatique "

présentée et soutenue publiquement par

David BUSSON

Le 26 Novembre 2018

Gestion de robots mobiles et collaboratifs en environnement contraint et dynamique

-

Management of mobile and collaborative robots in cluttered and dynamic environment

Directeur de thèse : **Richard BEAREE**
Co-encadrement de la thèse : **Olivier GIBARU et Adel OLABI**

Jury

M. Rachid ALAMI, Directeur de recherche, LAAS, Toulouse	Président
M. Vicent PADOIS, Maître de Conférence HDR, ISIR, Sorbonne Université, Paris	Rapporteur
Mme Hélène CHANAL, Maître de Conférence, IFMA, Université Blaise Pascal, Clermont-Frd	Rapporteur
M. Bruno SICILIANO, Professeur, PRISMA Lab, Università Federico II, Naples	Examinateur
M. Richard BEAREE, Professeur, LISPEN, Arts et Métiers ParisTech, Lille	Examinateur
M. Olivier GIBARU, Professeur, LISPEN, Arts et Métiers ParisTech, Lille	Examinateur
M. Adel OLABI, Maître de Conférence, LISPEN, Arts et Métiers ParisTech, Lille	Examinateur
M. Thierry BAZIRE, Ingénieur R&D, KUKA, Bordeaux	Invité

T H È S E

Arts et Métiers ParisTech - Campus de Lille
LISPEN

2016-ENAM-0041

École doctorale n° 432 : Science des Métiers de l'ingénieur

Doctorat ParisTech

T H È S E

pour obtenir le grade de docteur délivré par

l'École Nationale Supérieure d'Arts et Métiers

Spécialité " Automatique "

présentée et soutenue publiquement par

David BUSSON

Le 26 Novembre 2018

Gestion de robots mobiles et collaboratifs en environnement contraint et dynamique

-

Management of mobile and collaborative robots in cluttered and dynamic environment

Directeur de thèse : **Richard BEAREE**
Co-encadrement de la thèse : **Olivier GIBARU et Adel OLABI**

Jury

M. Rachid ALAMI, Directeur de recherche, LAAS, Toulouse	Président
M. Vicent PADOIS, Maître de Conférence HDR, ISIR, Sorbonne Université, Paris	Rapporteur
Mme Hélène CHANAL, Maître de Conférence, IFMA, Université Blaise Pascal, Clermont-Frd	Rapporteur
M. Bruno SICILIANO, Professeur, PRISMA Lab, Università Federico II, Naples	Examinateur
M. Richard BEAREE, Professeur, LISPEN, Arts et Métiers ParisTech, Lille	Examinateur
M. Olivier GIBARU, Professeur, LISPEN, Arts et Métiers ParisTech, Lille	Examinateur
M. Adel OLABI, Maître de Conférence, LISPEN, Arts et Métiers ParisTech, Lille	Examinateur
M. Thierry BAZIRE, Ingénieur R&D, KUKA, Bordeaux	Invité

T H È S E

Arts et Métiers ParisTech - Campus de Lille
LISPEN

Management of Mobile and Redundant Manipulators in Cluttered and Dynamic Environment

- Applications to the Aircraft Production Industry -

Abstract

Industrial applications involving collaborative robots are regarded with a growing interest. These power-limited systems are embedded with additional sensing capabilities, which allow them to safely work around humans and conquer new industrial grounds. The subject of managing redundant, collaborative and mobile systems, for assembly operations within a human-populated aircraft production environment, is addressed in this thesis. From a process perspective, the use of these smaller and less stiff counterparts of the non-collaborative robots comes with new challenges. Their high mechanical flexibility and weak actuation can cause shortcomings in positioning accuracy or for interaction force sustainment. The ever-changing nature of human-populated environments also requires highly autonomous solutions. In this thesis, a formulation of positional redundancy is presented. It aims at simplifying the exploitation of the freedom redundant manipulators have on static-task-fulfilling postures. The associated formalism is then exploited to characterise and improve the deformational behaviour and the force capacity of redundant serial systems. Finally, the subject of planning motions within cluttered and dynamic environments is addressed. An adaptation of the well-known Probabilistic RoadMaps method is presented – to which obstacles trajectories anticipation has been included. This solution allows to plan safe, efficient and non-intrusive motions to a given destination.

Keywords : Industrial robotics, Redundancy, Robot accuracy, Force capacity, Motion planning, anticipation.

Contents

List of Figures

List of Algorithms

Introduction and Context

As in other fields, the aircraft production industry has seen major changes with the robotisation of its processes. While many operations are still performed by human operators, an increasing number of machines populate the shop floors to take over the most tedious or repetitive tasks. One of the most recognised advantage of using automated machines is their high repeatability which ensures a high quality of execution. The use of robotic arms is popularising in the industry because these tireless systems are flexible, can reach and work over large volumes, while their compactness allow them to be transported to where work needs to be done.

In the aircraft field, the robotised production is widely occupied by high payload systems, able to carry versatile and heavy (>200 kg) tools such as Kuka Systems Aerospace best selling product, a multi-function end-effector Fig. **??**. The use of these heavy robots also comes with drawbacks, as safety measures impose to surround them with physical fences preventing human workers to be in the vicinity. This constraint has been hindering the deployment of robotic solutions in human populated shop floor up until recently, with the arrival of collaborative robots. Collaborative robots are robotic systems that are tailored to work alongside humans [1]. They are generally embedded with additional sensors. A non exhaustive list of these sensor includes force and torque sensors for collision detection, cameras, distance-computing, laser sensors. These sensors allow them to detect and/or adapt to human presence. Another key characteristic of these safe robots is their limited power. This has consequences on their speed performances, and payload, which rarely exceeds 25 kg. A last important feature is their easy programmability, which is supposed to quicken their integration in existing industrial processes, and to make them more easily reprogrammed to new situations or processes. Additionaly to their safety related

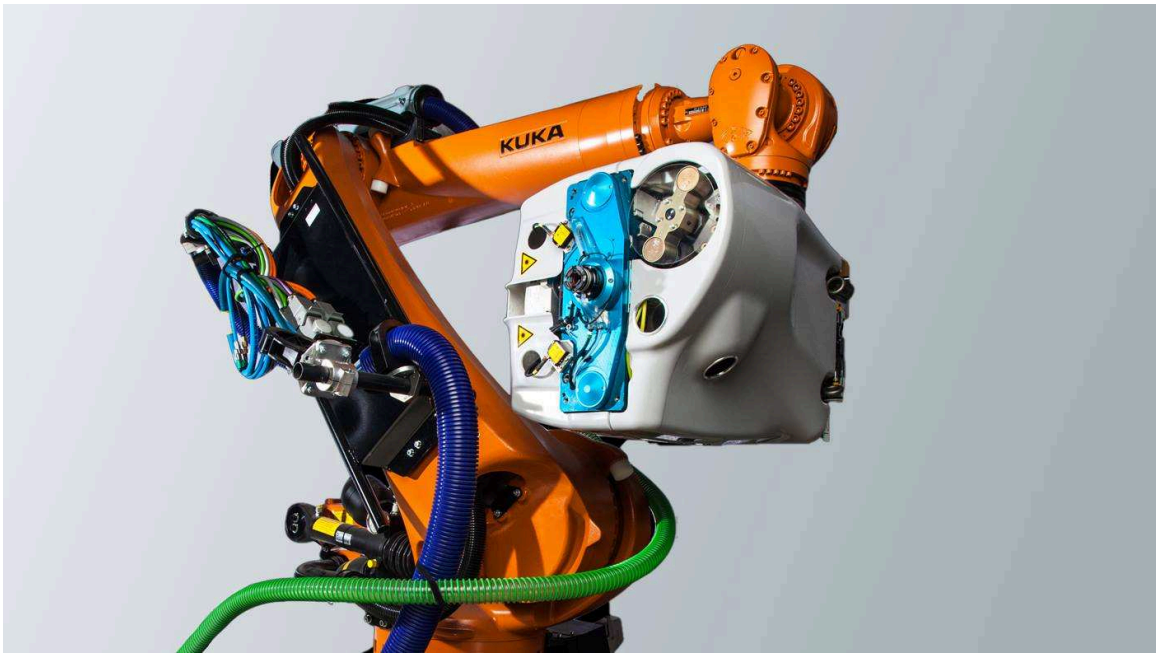


Figure 0.0.1: Multi-function end-effector produced by Kuka Systems Aerospace carried by a high payload robot (KR300).

features, the cobots are generally small robots, which make them able to squeeze in more cluttered environments, and bring more flexibility to the workplace.

System presentation

Nevertheless, transiting from high payload non collaborative systems to cobots comes with difficulties. The first one of these is the lower working volume issue, as cobots are generally much smaller systems than traditional robots. To match the reachability of the latter, company Kuka Systems Aerospace ordered the study of the integration of a flock of collaborative robot LBR iiwas, mounted on mobile platforms (KMP) to perform assembly tasks on Aircraft shopfloors (see Fig. 0.0.2). In addition to matching the work volume of traditional robots, the mobility and flock aspect bring a much desired flexibility, which is at the foundation of industry 4.0. These industrial

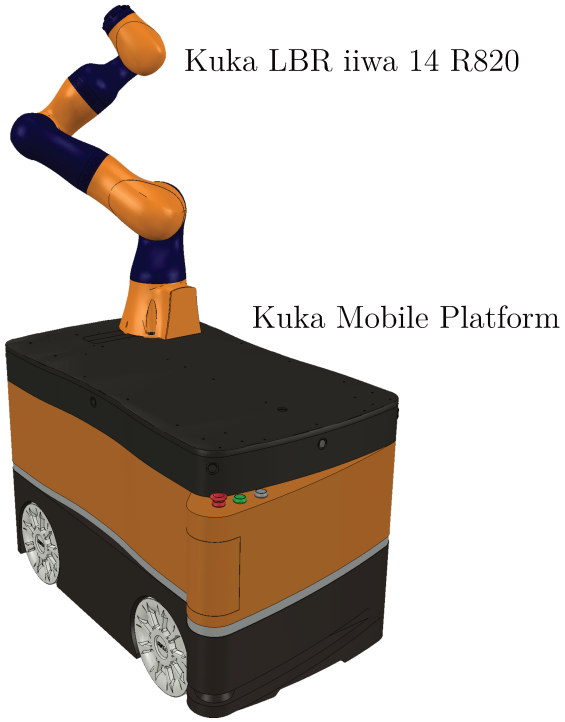


Figure 0.0.2: Kuka LBR iiwa 14 R820 mounted on Kuka Mobile Platform (KMP)

systems are designed for human collaboration and feature a number safety related capabilities. The serial robotic arm is a 29 kg, 7 rotary joints robot which is able to carry a 14 kg payload. The payload/weight ratio of this system makes it one of the most weight efficient industrial arm on the market. Additionally, the robot embeds accurate torque sensors at the output of each of its joints. These sensors impart the possibility to feel physical interactions happening on its links. They endow the system with very efficient contact sensing features, compliant interaction and hand-guiding capabilities. These abilities provide very efficient and reliable means of maintaining the mechanical integrity of the system itself. More importantly, they contribute to making the cohabitation and collaboration of humans and robots in the same vicinity safer. Furthermore, these abilities are very useful in assembly task

operations, where sensing and adapting to non perfect part positioning and shapes becomes more important than having a high end-effector positioning accuracy. The KMP is a 4-mecanum wheeled mobile platform. These wheels endow the platform with omnidirectional motion capabilities in the floor plane. This brings 3 additional degrees of freedom (DOFs) to the entire system, which add up to 10 DOFs. In terms of safety features, the platform possess two diagonally opposed 270°-ranged SICK laser scanners that monitor the surrounding physical objects situated at 15 cm above the floor level. The system carries its own battery, which can provide energy to the mobile base and the robotic arm for several consecutive hours.

Industrial locks

In 2016, Airbus organised at the International Conference of Robotics and Automation (ICRA), Stockholm, a competition aiming at using lightweight robots ($<$ 100 kg) to drill aircraft quality holes into a 70×70 cm^2 aluminium panel (Fig. 0.0.3). Unremarkably, the marking criteria were related to the cycle time (60 minutes to drill 245 holes), accuracy on hole positions (tolerance of $\pm$ 0.5 mm) and hole quality (good panel surface condition), which are some of the main quality indicators of the robotised production field. For this competition, we chose to use the LBR iiwa alone, mounted with a drilling end-effector, itself endowed with a vision and laser-based localisation system and a lubrication tool device (see Fig. 0.0.4).

This challenge was certainly a turning point of this PhD, and the main problems encountered within the preparation for and during the competition have had a strong influence over the organisation of this thesis.

The first of them, which was encountered in the preparation of the challenge, was to choose a suitable placement of the robot base, which enabled reachability of the widest possible number of holes. During the challenge, despite an accurate vision-based localisation method, unexplained holes positioning misalignments were

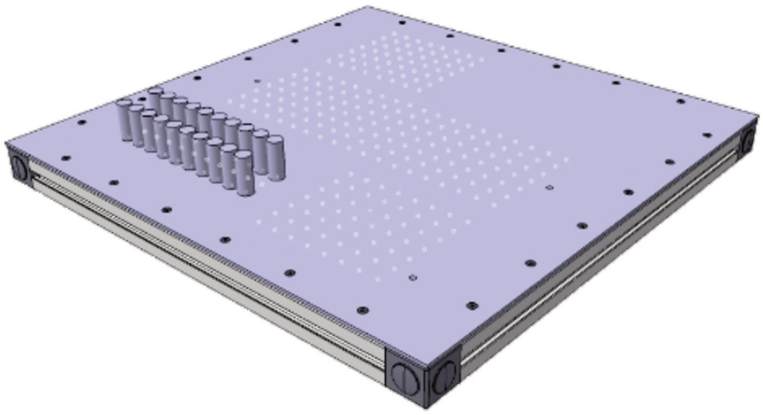


Figure 0.0.3: Airbus Shopfloor Challenge objective panel.

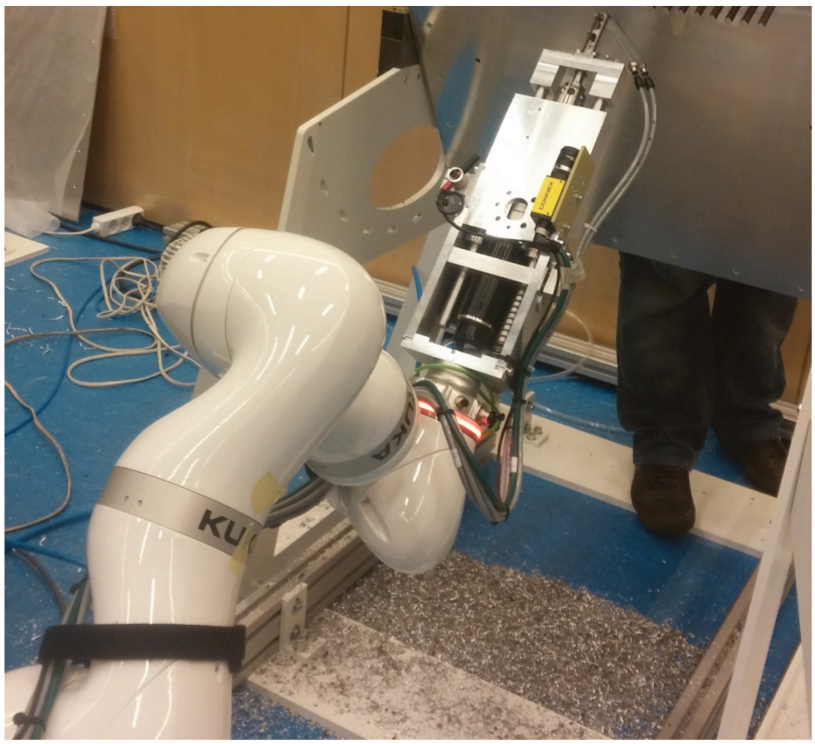


Figure 0.0.4: 7-joints LBR iiwa and drilling end-effector during the Airbus Shopfloor Challenge, ICRA 2016, Stockholm.

spotted. We later understood that these inaccuracies were strongly influenced by the mechanical deformation of the serial system under the stress of the the drilling procedure. In parallel, we witnessed, without any easy means to predict or avoid it, that the robot torque limits were sometimes reached before a hole was completely drilled. The uneven distribution of these unforeseen misalignments or torque shortage occurrences over the drilling panel tipped us into thinking that the posture used by the serial system had a strong influence on the progress of the drilling operation. In the competition, the 70×70 cm^2 panel was to be drilled by 245 holes. Densely packed as they were, moving from one to an other in a straight line end-effector path regularly led near singular configurations. The Cartesian controller of the robot prevented the robot from performing these motions and therefore many holes were missed. This happened despite the existence of an associated articular configuration within joint bounds for the hole location. We made the decision to use articular motions, *i.e.* motions interpolating linearly in joint space between the current articular position and the articular position destination leading the end-effector in the right location. This led to quite unpredictable motions of the end-effector and the occurrence of collisions with the environment, which are unacceptable in an industrial context.

This experience led us to orientate the PhD work towards improving our geometric and kinematic understanding of the system. This allowed us in time to characterise the inaccuracies caused by mechanical flexibility, and the force capacity of the system with regards to a specific task. Strategies were devised in order to minimise the influence of these shortcomings by exploiting the task redundancy of the system.

Thesis contents and organisation

A simple analogy can be used to describe the way autonomous industrial processes in dynamic and cluttered environment can be decomposed for mobile and redundant systems. The human knocking in nails in Fig. 0.0.5 uses a similar strategy. The core steps can be extracted to provide an abstract description of the procedure. At the beginning, the human is given a task to put a nail somewhere in the wall. The robotic

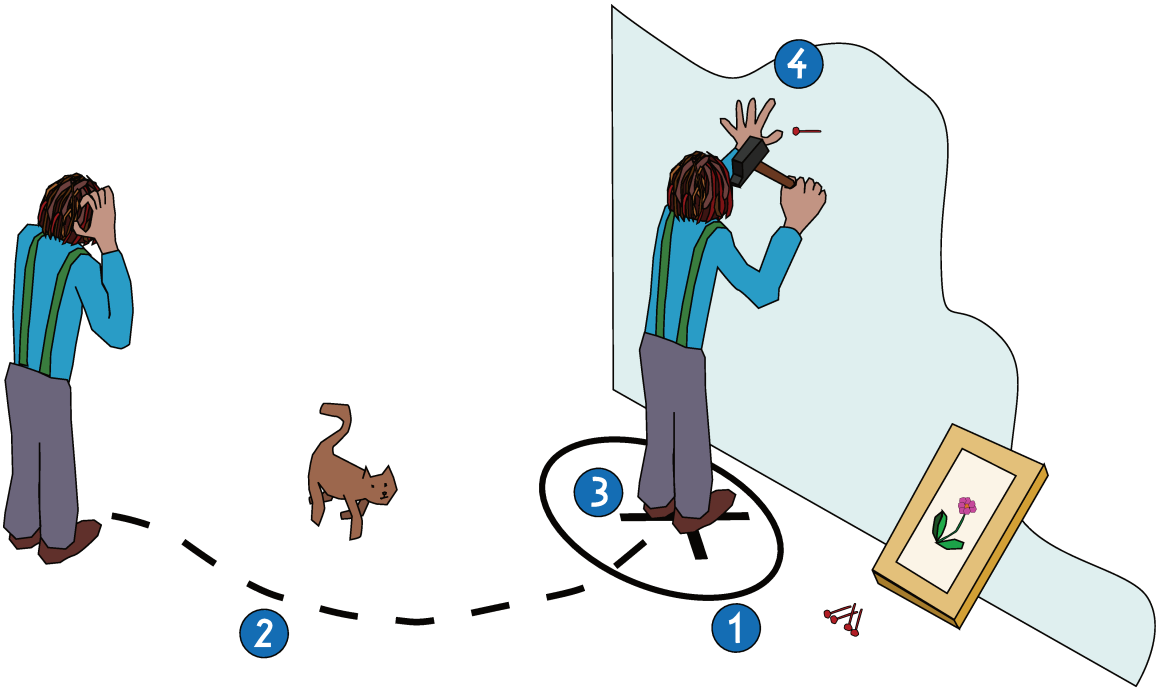


Figure 0.0.5: A simple analogy used to extract to core steps a mobile and redundant robotic system can follow to autonomously fulfil an industrial task.

system is also given a task, that can be described geometrically.

1. From this task, our human determines roughly where to position its feet to allow reachability of the nail position. The system determines for its base an accessible area for which any position enables it to place its end-effector according to the task definition.

2. The human then needs to move to his chosen area without colliding with furniture, walls, or his pet. The industrial system needs to efficiently move to its chosen area. Its environment may be populated with static or moving obstacles, which need to be safely avoided.

3. After roughly reaching his destination, the human could choose any placement for his feet and try to drive the nail in the wall. However, some postures don't allow him to do so in good conditions, despite enabling reachability of the task.

Knocking the nail while turning his back to the wall could lead to poor results in terms of efficiency, accuracy, or just by preventing him from applying enough strength to the task. One type of posture is generally admitted to be the correct one to knock in nails, this is the one he chooses. The same applies to the system, which needs to decide of a suitable posture to perform the task. Our system is redundant, and could position its end-effector correctly with many different postures. One needs to be chosen that allows it to perform the process in good conditions.

4. After choosing a suitable posture, the human can start driving the nail in the wall. The industrial system can also start performing its process when placed in a suitable posture.

The system described in the previous section, which is composed of a robotic arm and a mobile platform, can be synthesised into 10 independent DOFs leading to a displacement of the terminal link. This kinematic structure possesses 4 DOFs that traditional 6-DOFs industrial robots don't have. A major challenge associated to introducing this redundancy in an industrial context is to retain the advantage of flexibility and dexterity while making the management of these extraordinary kinematic capabilities as simple and intuitive as possible. The enlightened management of these tremendous yet complex capabilities is the purpose of Chapter 1. The results presented therein serve as a foundation to the work presented in the two following chapters.

Despite their undeniable attractiveness, the use of these robots for processes that are traditionally done by their strong industrial counterparts comes with highly complex challenges. Cobots are generally lighter than their non collaborative counterparts and composed of parts that are proportionally thinner. Consequently, the mechanical components have lower mechanical stiffnesses, and easily deform under the strain caused by interacting with the environment. This sometimes causes unacceptable accuracy issues. In this thesis, a strategy is presented in Chapter 2 to cope with the low mechanical stiffness and with the deformations arising from the interactions of

cobots with their environment.

To abide to safety specifications, collaborative robots can transmit a very limited amount of power, which translates into rather low motor torques and speeds. Consequently, robot manufacturers try to make these systems as light as possible, as inertial and gravity compensation of the robot links tend to account for a large portion of their torque capacity. Despite this, cobots sometimes lack the motor torques needed to perform some operations. New strategies must be found to adapt to processes requiring high forces and torques capacities. Chapter 3 introduces the Force Capacity Index (FCI), which assesses the capacity of a serial robot to produce a given spatial force. The exploration of the self motion of kinematically redundant systems is then done to ensure a given force capacity.

Another subject that is tackled within this PhD work is the problem of how to efficiently and safely move within an unstructured and dynamic environment that is populated by humans and robots alike. Indeed, while robots are able to perform thousands of times the exact same motion within a geometric range no human could hope to match, they are not intelligent, and thus unable to adapt to new situations. Challenges arising with the use of systems designed for repeatability, in an ever-changing environment are also numerous. The presence of humans greatly complicates the management of the displacements of mobile collaborative systems. One of these challenges, which is approached in Chapter 4 is the safe and efficient motion planning of these systems in dynamic and cluttered environments.

Thesis positioning and contributions

The overall positioning and organisation of the thesis is reminded on Fig. 0.0.6.

The second chapter is inspired from the author's publication at the International Federation of Automatic Control (IFAC) [2], which took place in Toulouse, 2017. The third chapter is largely copied from the author's publication at the Robotics and Automation Letters (RA-L) and conference paper at the International Conference on

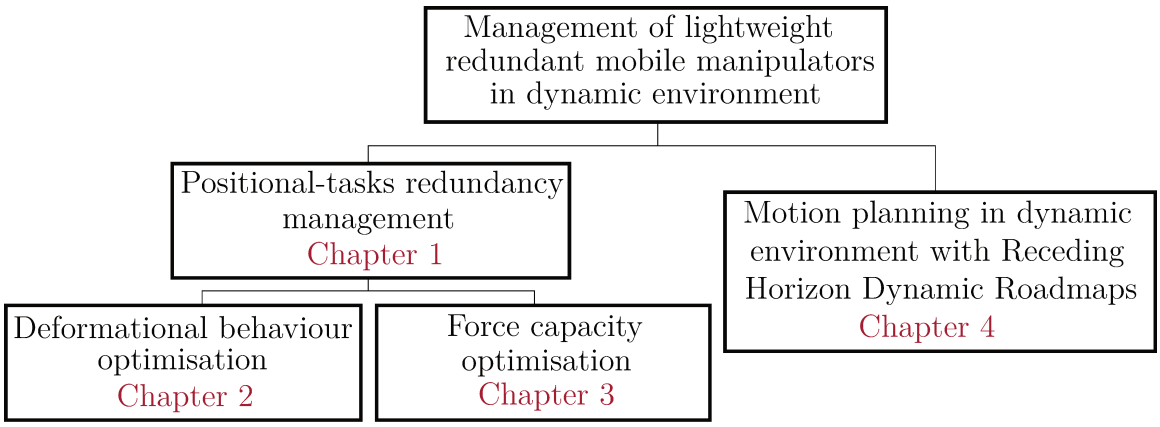


Figure 0.0.6: Thesis positioning and organisation.

Robotics and Automation (ICRA) [3], which took place in Brisbane, 2018.

1

An enlightened redundancy management

Kinematic redundancy is a vast subject and is dealt with through various strategies. The chapter begins by introducing the way redundancy is traditionally used and delt with, with inverse differential schemes in Section 1.2. Then, Section 1.3 introduces the framework of redundancy spaces, that formalises the concept of redundancy at the position level. Positional tasks are commonly assigned operations in the robotised production industry. Pick and place operations, riveting, screwing, drilling, or local measurements are among these so called positional tasks, because their associated geometric specifications can be given in terms of static end-effector locations. Redundant robots with regards to these tasks are able to comply with them in continuous manifolds of articular positions. This framework answers a need for simplifying and formalising the exploitation of the freedom redundancy offers at the position level. When it can be used, advantages of this framework, compared to differential kinematics methods, are numerous for the problem of choosing a configuration that complies with a positional task. It allows a thorough exploitation of geometric redundancy. Redundancy resolution using this framework are made simpler, lighter, more accurate and more exhaustive. This framework will be used in following chapters (Chapter 2 and Chapter 3) as a fundamental tool for the enhancement of posture-dependant performance characteristics of redundant robots.

Contents

1.1 Introduction to robot kinematics

1.1.1 Geometric nomenclature

The position and orientation of a rigid body in space can be grouped under the term *pose*. The minimum number of parameters needed to fully describe the pose of a rigid body in Euclidean space is six. A *coordinate reference frame* or *frame* $\mathcal{F}_i$ consists of an origin, denoted O_i, and a triad of mutually orthogonal basis vectors, denoted $(\mathbf{x}_i, \mathbf{y}_i, \mathbf{z}_i)$. The pose of a body is always expressed relative to some other body, and can be expressed as the pose of one coordinate frame relative to another. In the rest of the thesis, rigid body transformations will be based upon the homogeneous transformation matrix representation. The homogeneous transformation ${}^i\mathbf{T}_j$ leading frame $\mathcal{F}_i$ to frame $\mathcal{F}_j$, which is the same as the one describing the pose of frame $\mathcal{F}_j$ with respect to frame $\mathcal{F}_i$, is the 4×4 matrix :

$$ {}^i\mathbf{T}_j = \begin{bmatrix} & {}^i\mathbf{R}_j & & {}^i\mathbf{p}_j \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

.

${}^i\mathbf{R}_j$ can be seen as the 3×3 orientation matrix of frame $\mathcal{F}_j$ expressed in frame $\mathcal{F}_i$. ${}^i\mathbf{p}_j$ can be seen as the position of the origin of frame $\mathcal{F}_j$ expressed in frame $\mathcal{F}_i$.

${}^i\mathbf{R}_j$ can be otherwise written :

$$ {}^i\mathbf{R}_j = \begin{bmatrix} {}^i\mathbf{x}_j & {}^i\mathbf{y}_j & {}^i\mathbf{z}_j \end{bmatrix} = \begin{bmatrix} \mathbf{x}_j.\mathbf{x}_i & \mathbf{y}_j.\mathbf{x}_i & \mathbf{z}_i.\mathbf{x}_i \\ \mathbf{x}_j.\mathbf{y}_i & \mathbf{y}_j.\mathbf{y}_i & \mathbf{z}_i.\mathbf{y}_i \\ \mathbf{x}_j.\mathbf{z}_i & \mathbf{y}_j.\mathbf{z}_i & \mathbf{z}_i.\mathbf{z}_i \end{bmatrix} $$

where ${}^i\mathbf{x}_j$, ${}^i\mathbf{y}_j$, and ${}^i\mathbf{z}_j$ of ${}^i\mathbf{R}_j$ are the expressions of vectors $\mathbf{x}_j$, $\mathbf{y}_j$ and $\mathbf{z}_j$ in frame $\mathcal{F}_i$, which can be found through the dot product of the vectors of the two coordinate frames.

1.1.2 Task Space Formulation

For a serial robot consisting of N rigid bodies, a minimum set of $6N$ independent coordinates is necessary to fully describe the position and orientation of all the bodies relative to a reference frame. Since the links of the robot are connected together, there exist constraint equations that geometrically bound these coordinates between each others. Considering this, these $6N$ coordinates can be expressed in terms of a smaller set of N coordinates gathered in a vector $\mathbf{q}$, along with a non-varying geometric description of the geometric relations between these links (see DH parameterisation in Section 1.1.3). These N coordinates are called generalised coordinates, and are fully independent from one another.

The articular position coordinates, gathered in a $\mathbf{q} = \begin{bmatrix} q_1 & \dots q_N \end{bmatrix}^\intercal$ vector for a N-DOFs robot, are such a set of generalised coordinates. They are used to naturally and efficiently describe an entire robot posture. This articular position vector, also termed *configuration*, is part of an articular space, also called *joint space*. Basic robot control handles generally include these joint coordinates or their derivatives with respect to time. Queries that are specified in terms of trajectories of these articular coordinates are generally understood by the system:

> "*Move from this articular configuration to that one using a bang bang limited jerk profile interpolating along a straight path (in configuration space) joining the two configurations with non-zero final velocity, then keep a constant articular velocity for two seconds and finally stop as quickly as you can along a straight path in articular space*"

Unfortunately, these handles are generally not easily intuited by robot users, who generally prefer to specify their orders with more abstract notions.

A robot is in essence a system that humans use to perform actions. From the user's point of view, these actions/tasks/operations are generally formulated in what are considered abstract terms for a robot:

> *"Translate your end effector to this location while avoiding obstacles with the rest of your links with a smooth bell-shaped Cartesian velocity profile,*

then follow as best as possible a moving frame in position and orientation with your referenced tool tip for 2 seconds, then apply this force with your tool tip at this location...”

The mathematical description of a task may be gathered in a vector $\mathbf{t} = \begin{bmatrix} t_1 & \dots t_M \end{bmatrix}^\intercal$, termed the *task vector*, or *operational vector*. Very often, this vector describes the pose of a frame, which is attached to the end effector of a robot, expressed in a reference frame. The task is not always as contraining as that, and may contain just a position or an orientation, or any degratation of a pose. On the other hand, it can be more contraining, if it contains several poses of subelements of the robot. Each component t_i of the vector contains a single abstract information about the action to perform. The operational vector is itself part of an *operational space* or *task space*.

1.1.3 System geometric description

Cartesian positioning and trajectory following of an end-effector are among the most prevalent primitive tasks that robots are assigned to perform, especially from the perspective of an industrial robot programmer. These specific tasks are so common in traditional robot uses that, in the roboticits semantics, the term ”task” often amalgamates to these primitives only.

The first difficulty lies in translating these high level needs into the control handles of robotic systems. The field of serial robot *kinematics* relates to this problem. Translating joint space coordinates into end-effector space localisation coordinates is called the *direct modelling* of a robotic system. When the coordinates relate to the position of the end effector, we talk about the *direct geometric model*. When they relate to the speed at which the end effector moves, we talk about the *direct kinematic model*.

The geometric description of serial manipulators can be conveniently and efficiently done with one of the numerous adaptations of the Denavit and Hartenberg fundamental convention [4]. Among these adaptations, one of the most wildly used so called *DH description* or *parameterisation* is the one defined by Khalil and Kleinfinger in [5]. The parameters notations will be the ones defined in [6]. This latter convention uses four parameters to describe efficiently and without ambiguity the geometry of a

rigid kinematic chain. It can be defined recursively for any open or closed loop system through the judicious placement of the reference frames origin and axes of the links composing the chain. A rule is to have the $\mathbf{x}$ axis of one frame intersects and being perpendicular to the $\mathbf{z}$ axis of the preceding reference frame.

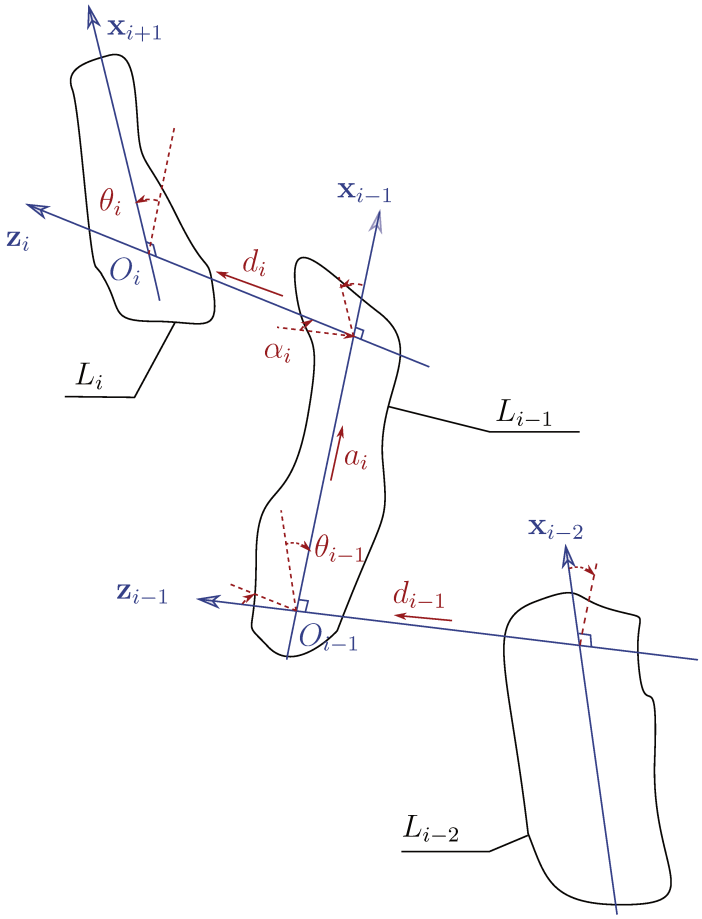


Figure 1.1.1: Modified DH graphic explanations.

For a serial robot composed of N joints and $N+1$ links, numbered from 0 to N, the parameters of the DH description are defined for each link i ($i \in [\![1..N]\!]$), as:

- the link length a_i, which is the distance from $\mathbf{z}_{i-1}$ to $\mathbf{z}_i$ along $\mathbf{x}_{i-1}$,
- the link twist α_i, which is the angle from $\mathbf{z}_{i-1}$ to $\mathbf{z}_i$ about $\mathbf{x}_{i-1}$,

- the joint offset d_i, which is the distance from $\mathbf{x}_{i-1}$ to $\mathbf{x}_i$ along $\mathbf{z}_i$,
- the joint angle θ_i, which is the angle from $\mathbf{x}_{i-1}$ to $\mathbf{x}_i$ about $\mathbf{z}_i$,

This convention has the advantage to relatively locate two successive reference frames - frame i relative to frame $i-1$ - thanks to the composition of four simple operations. Frame i is located relative to frame $i-1$ by first executing a rotation by α_i about the $\mathbf{x}_{i-1}$ axis. Then a translation of a distance a_i along $\mathbf{x}_{i-1}$ is to be performed. After that, comes a rotation by θ_i about the $\mathbf{z}_i$ axis. Finally, a translation along $\mathbf{z}_i$ by d_i is to be carried out. This composition can be written in terms of an homogeneous transformation, as :

$$ {}^{i-1}\boldsymbol{T}_i = \begin{pmatrix} \cos(\theta_i) & -\sin(\theta_i) & 0 & a_i \\ \sin(\theta_i)\cos(\alpha_i) & \cos(\theta_i)\cos(\alpha_i) & -\sin(\alpha_i) & -\sin(\alpha_i)d_i \\ \sin(\theta_i)\sin(\alpha_i) & \cos(\theta_i)\sin(\alpha_i) & \cos(\alpha_i) & \cos(\alpha_i)d_i \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{1.1} $$

1.1.4 Direct modelling of robotic systems

1.1.4.1 Direct geometric model for positioning tasks

The direct geometric model of a kinematic chain corresponds to the mathematical relations calculating the *operational coordinates* of the system when it is set in an articular position. The operational coordinates may represent various physical quantities, but the *Direct Geometric Model* (or *Forward Geometric Model*) often relates to operational coordinates representing the pose of a frame which is attached to a tool of a N-DOFs system. This pose can be represented by an homogeneous transformation matrix ${}^0T_N$ representing the extremal link reference frame $\mathcal{F}_N$ pose relative to the robot base reference frame $\mathcal{F}_0$. ${}^0T_N$ can be written as the composition of the elementary homogeneous matrices derived from the DH parameterisation of the system as :

$$ {}^0\mathbf{T}_N = {}^0\mathbf{T}_1 \dots {}^{N-1}\mathbf{T}_N = \prod_{i=1}^{N} {}^{i-1}\mathbf{T}_i $$

Thanks to Eq. (1.1) and to the DH parameterisation of the system, the expressions for each of these transformation matrices ${}^{i-1}\mathbf{T}_i$, as well as the expression of ${}^{0}\mathbf{T}_N$ can be fully derived.

1.1.4.2 Task Jacobian matrices

The direct geometric model has given us a relation between the articular position coordinates of a system and its operational position coordinates related to the pose of a frame attached to the robot's end-effector, thus creating a mapping from joint space to task space. The mapping is said to have been performed at the position level. It takes the form:

$$\mathbf{t} = \mathbf{F_t}(\mathbf{q}) \tag{1.2}$$

where $\mathbf{t}$ is the vector containing the operational space coordinates, $\mathbf{F_t}$ is a potentially non-linear vector function of $\mathbf{q}$, which is the vector containing the articular coordinates. It is interesting to consider the first order differential kinematic relation:

$$\dot{\mathbf{t}} = \boldsymbol{J}_t(\mathbf{q})\dot{\mathbf{q}} \tag{1.3}$$

which can be obtained by differentiating Eq. (1.2) with respect to time. In this expression, $\dot{\mathbf{t}}$ is the operational space velocity, $\boldsymbol{J_t}(\mathbf{q})$ is called the *operational space Jacobian*, *task Jacobian* or *analytic Jacobian* matrix [1], and $\dot{\mathbf{q}}$ is the articular velocity vector. The Jacobian matrix is built thanks to the partial derivative of the component of the vector value function $\mathbf{F_t}$ by each element of the articular coordinate vector $\mathbf{q}$. As such, a convenient, yet mathematically incorrect notation, denotes the Jacobian matrix as : $\boldsymbol{J_t} = \partial\mathbf{F_t}/\partial\mathbf{q}$.

The analytical Jacobian matrix is the representation of the operational space velocities with respect to the articular velocities in a given articular configuration of the system. Performing an order one approximation of the Taylor expansion of the ith component $F_{\mathbf{t}_i}$ of the direct geometry function $\mathbf{F_t}$ defined in Eq. (1.2) leads to a

[1] In the rest of the thesis, a Jacobian matrix will indifferently be called "Jacobian matrix", or merely "Jacobian"

mathematical formulation of the ith component δt_i, of the task displacement $\boldsymbol{\delta}\mathbf{t}$, with regards to a small variation of articular configuration $\boldsymbol{\delta}\mathbf{q} = [\delta q_1, \dots, \delta q_m]^\intercal$.

$$
\begin{aligned}
F_{\mathbf{t}_i}(\mathbf{q}+\boldsymbol{\delta}\mathbf{q}) &= F_{\mathbf{t}_i}(\mathbf{q}) + \sum_{j=1}^{m} \frac{\partial F_{\mathbf{t}_i}(\mathbf{q})}{\partial q_j}\delta q_j + \mathcal{O}(\|\boldsymbol{\delta}\mathbf{q}\|^2) \\
\Rightarrow \mathbf{F}_\mathbf{t}(\mathbf{q}+\boldsymbol{\delta}\mathbf{q}) &= \begin{bmatrix} F_{\mathbf{t}_1}(\mathbf{q}) \\ \vdots \\ F_{\mathbf{t}_m}(\mathbf{q}) \end{bmatrix} + \boldsymbol{J}_\mathbf{t}\boldsymbol{\delta}\mathbf{q} + \begin{bmatrix} \mathcal{O}(\|\boldsymbol{\delta}\mathbf{q}\|^2) \\ \vdots \\ \mathcal{O}(\|\boldsymbol{\delta}\mathbf{q}\|^2) \end{bmatrix} \\
\Rightarrow \boldsymbol{\delta}\mathbf{t} = \mathbf{F}_\mathbf{t}(\mathbf{q}+\boldsymbol{\delta}\mathbf{q}) - \mathbf{F}_\mathbf{t}(\mathbf{q}) &= \boldsymbol{J}_\mathbf{t}\boldsymbol{\delta}\mathbf{q} + \begin{bmatrix} \mathcal{O}(\|\boldsymbol{\delta}\mathbf{q}\|^2) \\ \vdots \\ \mathcal{O}(\|\boldsymbol{\delta}\mathbf{q}\|^2) \end{bmatrix} \\
\Rightarrow \boldsymbol{\delta}\mathbf{t} &\approx \boldsymbol{J}_\mathbf{t}\boldsymbol{\delta}\mathbf{q}\ . \qquad (1.4)
\end{aligned}
$$

For a small variation of articular configuration $\boldsymbol{\delta}\mathbf{q}$, the task displacement $\boldsymbol{\delta}\mathbf{t}$ can be approximated at articular configuration $\mathbf{q}$ thanks to Eq. (1.4).

The Jacobian matrices can be seen as representations of the operational space capabilities of the system in a given articular configuration. The word *mapping* between articular and operation spaces is often used to stress this notion. The system's ability to produce operational velocities is directly related to the components of these matrices. By definition, column i of a Jacobian matrix relates to the influence of the positional variations of joint i onto all operational space coordinates.

In some configurations, the task Jacobian matrix may become rank-deficient. Looking at the definition of this Jacobian matrix through the expression $\boldsymbol{J}_\mathbf{t}(\mathbf{q}) = \partial\mathbf{F}_\mathbf{t}/\partial\mathbf{q}$, it becomes clear that rank deficiency translates into disabling certain directions of the operational space. These configurations are called *singular*. The analysis of the Jacobian through a *Singular Value Decomposition (SVD)* [7, 8] can help avoid these singular configurations where the manipulator looses or is about to loose operational mobility and compensates by changing very sharply its articular configuration (more

information will be provided on this subject in Section 1.2.2).

Jacobian Matrices are very useful in robotics as they are at the base of the inverse differential model, which can be used to compute the articular velocities $\dot{\mathbf{q}}$ needed to comply with operational velocities $\dot{\mathbf{t}}$. Jacobian matrices can also be used to compute a local solution $\mathbf{q}$ of joint coordinates corresponding to an operational position $\mathbf{t}$ through numerical integration techniques (numerical methods such as Newton Raphson and gradient descent are explained in [9]).

1.1.4.3 Geometric Jacobian matrix

One particular Jacobian, which is of major interest in the kinematic analysis of manipulators is called the *geometric Jacobian*. This Jacobian matrix is a linear mapping between the articular velocities of the system and the *spatial velocity* of the end effector. Other closely related formalisms for spatial velocity include the *twist* [10, 11] or its French version the *torseur cinématique* [12](which we will call "torseur" for commodity). When the operational coordinates relate entirely to locating the end effector of the system in position and orientation, the inversion of analytic Jacobians may suffer from *representation singularities* (or *algorithmic singularities*), that originate from the representation of the orientation [13]. The geometric Jacobian then becomes a very powerful tool, because the representation of velocities used therein don't suffer from representation singularities. Therefore, a configuration may be singular for an analytic Jacobian while it is not for another that uses another representation of orientations. The geometric Jacobian doesn't suffer from algorithmic singularities, and a loss of rank of this Jacobian always translates into a *kinematic singularity*, *i.e.* a loss of mobility in space.

The formalism of "torseurs cinématiques", or "torseurs" can be used to express the geometric Jacobian of a N-DOFs serial robot. Within the torseur notation, rigid bodies will sometimes be referred to with the frames that are attached to them (*e.g* S_i might be referred to as "$\mathcal{F}_i$" or just "i"). Given a point A which can be located anywhere in space, the torseur $\{\mathcal{V}_{S_i/j}\}^A$ in point A of a rigid body S_i relative to a

coordinate frame $\mathcal{F}_j$ is constructed from a pair of 3-dimensional vectors, the angular velocity and the linear velocity

$$\{\mathcal{V}_{S_i/j}\}^A = \begin{Bmatrix} \mathbf{\Omega}_{S_i/j} \\ \mathbf{V}_{A\in S_i/j} \end{Bmatrix} . \tag{1.5}$$

The linear velocity will be noted $\mathbf{V}_{A\in S_i/j}$. Given a rigid body S_i, it expresses the linear velocity of the body-fixed (*i.e.* "S_i-fixed") point which currently coincides with point A, with regard to frame $\mathcal{F}_j$. The angular velocity of solid S_i with regard to frame $\mathcal{F}_j$ will be noted $\mathbf{\Omega}_{S_i/j}$. It expresses the rate of change of any vector that is attached to solid S_i with regard to $\mathcal{F}_j$. Let us define $\{\mathcal{V}_{N/0}\}^P$, the torseur in point[2] P of link N, the extremal link of the robot, relative to $\mathcal{F}_0$, the robot base reference frame. The law of relative motions applied to the serial system gives

$$\begin{aligned} \{\mathcal{V}_{N/0}\}^P &= \{\mathcal{V}_{N/N-1}\}^P + \{\mathcal{V}_{N-1/N-2}\}^P + \cdots + \{\mathcal{V}_{2/1}\}^P + \{\mathcal{V}_{1/0}\}^P \\ &= \sum_{i=1}^{N} \{\mathcal{V}_{i/i-1}\}^P , \end{aligned} \tag{1.6}$$

where

$$\{\mathcal{V}_{i/i-1}\}^P = \begin{Bmatrix} \mathbf{\Omega}_{i/i-1} \\ \mathbf{V}_{P\in i/i-1} \end{Bmatrix} . \tag{1.7}$$

If joint i is a revolute joint:

$$\mathbf{\Omega}_{i/i-1} = \dot{q}_i \mathbf{z_i} \tag{1.8}$$

where $\dot{q}_i$ is the derivative of q_i, joint i angular position, by time.

If joint i is a prismatic joint:

$$\mathbf{\Omega}_{i/i-1} = \mathbf{0} \tag{1.9}$$

The linear velocity $\mathbf{V}_{P\in i/i-1}$ of the body-fixed point that currently coincides with P

[2]This point is often conveniently chosen to be the origin of the TCP reference frame, which is generally attached to end effector of the system.

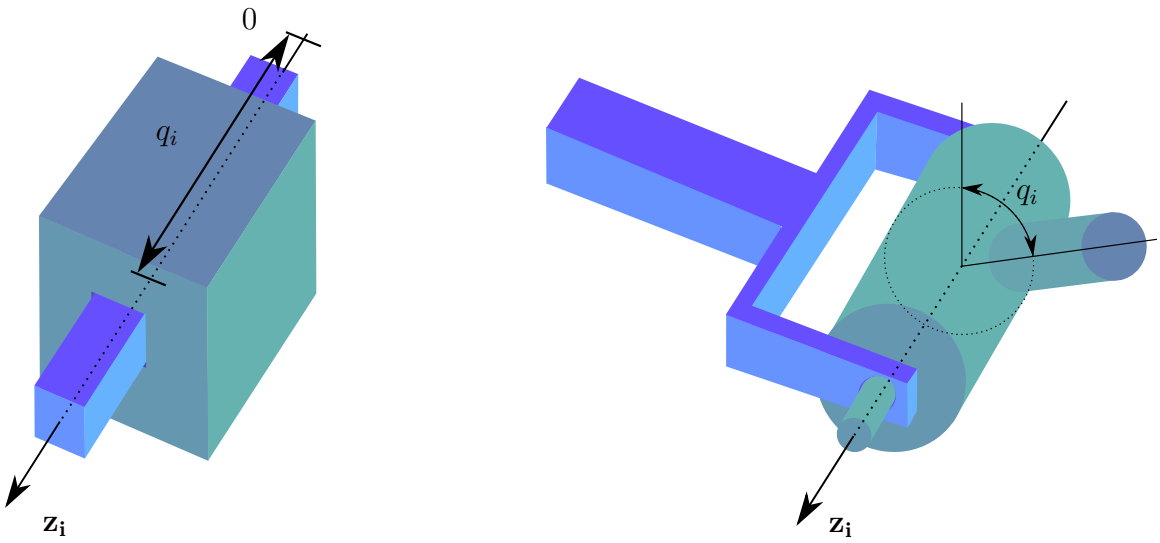


Figure 1.1.2: A prismatic and a revolute joints.

and which is rigidly attached to link i, relative to frame $\mathcal{F}_{i-1}$ (which is itself attached to link $i-1$) can be computed thanks to the relation of moment transfers (Varignon's theorem) [14] on a point situated on the rotary axis of joint i. Such points can be found through the DH parameterisation of the robot, taking the origin O_i of $\mathcal{F}_i$ which is attached to the link i:

$$\mathbf{V}_{P\in i/i-1} = \mathbf{V}_{O_i\in i/i-1} + \mathbf{PO_i} \wedge \mathbf{\Omega}_{i/i-1}. \tag{1.10}$$

If joint i is revolute, in Eq. (1.10), the term $\mathbf{V}_{O_i\in i/i-1}$ is zero because any point situated on the rotary axis of joint i has a zero linear speed with regard to $\mathcal{F}_{i-1}$. Therefore, for revolute joints, $\mathbf{V}_{P\in i/i-1}$ simplifies into:

$$\mathbf{V}_{P\in i/i-1} = \mathbf{PO_i} \wedge \dot{q}_i\mathbf{z_i} \ . \tag{1.11}$$

If joint i is prismatic, the term $\mathbf{V}_{O_i\in i/i-1}$ corresponds to the translation velocity of the prismatic joint : $\dot{q}_i\mathbf{z_i}$. According to Eq. (1.9), $\mathbf{\Omega}_{i/i-1}$ is zero. Therefore, for

prismatic joints, $\mathbf{V}_{P\in i/i-1}$ simplifies into:

$$\mathbf{V}_{P\in i/i-1} = \dot{q}_i \mathbf{z_i} \ . \tag{1.12}$$

The geometric Jacobian $\boldsymbol{J}$ of the system in P can be seen as the columnwise concatenation of the differentiation of the torseur $\{\mathcal{V}_{0/N}\}^P$ by each joint velocity $\dot{q}_j$ (with $j \in [\![1..N]\!]$). The expression of the j$^{\text{th}}$ column of J is

$$\begin{aligned} \boldsymbol{J}_j &\equiv \frac{\partial\{\mathcal{V}_{0/N}\}^P}{\partial\dot{q}_j} \\ &\equiv \frac{\partial(\sum_{i=1}^{N}\{\mathcal{V}_{i/i-1}\}^P)}{\partial\dot{q}_j} \\ &\equiv \sum_{i=1}^{N}\frac{\partial\{\mathcal{V}_{i/i-1}\}^P}{\partial\dot{q}_j} \end{aligned} \tag{1.13}$$

Eq. (1.7), 1.8, 1.9, 1.11 and 1.12 reveal that $\{\mathcal{V}_{i/i-1}\}^P$ only depends on the articular velocity $\dot{q}_i$, vector $\mathbf{PO_i}$ and vector $\mathbf{z_i}$.

For revolute joints, J_j simplifies into

$$\begin{aligned} \boldsymbol{J}_j &\equiv \frac{\partial\{\mathcal{V}_{j/j-1}\}^P}{\partial\dot{q}_j} \\ \boldsymbol{J}_j &= \begin{bmatrix} \frac{\partial\dot{q}_j\mathbf{z_j}}{\partial\dot{q}_j} \\ \frac{\partial(\mathbf{PO_j}\wedge\dot{q}_j\mathbf{z_j})}{\partial\dot{q}_j} \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{z_j} \\ \mathbf{PO_j}\wedge\mathbf{z_j} \end{bmatrix}. \end{aligned} \tag{1.14}$$

For prismatic joints, J_j simplifies into:

$$\begin{aligned} \boldsymbol{J}_j &\equiv \frac{\partial\{\mathcal{V}_{j/j-1}\}^P}{\partial\dot{q}_j} \\ \boldsymbol{J}_j &= \begin{bmatrix} \frac{\partial\mathbf{0}}{\partial\dot{q}_j} \\ \frac{\partial\dot{q}_j\mathbf{z_j}}{\partial\dot{q}_j} \end{bmatrix} \\ \boldsymbol{J}_j &= \begin{bmatrix} \mathbf{0} \\ \mathbf{z_j} \end{bmatrix}. \end{aligned} \tag{1.15}$$

Now noticing that all the vectors composing $\boldsymbol{J}_j$ can be derived from the direct geometric model for positioning tasks (see Section 1.1.4.1), the geometric Jacobian can be computed. Often, the geometric Jacobian is expressed in the robot base reference frame, and sometimes, in the robot tool reference frame. It can also be given in any known reference frame by expressing $\mathbf{z_j}$ and $\mathbf{PO_j}$ in the corresponding frame. Each corresponding Jacobian expression are mappings from articular space displacements (or velocities), to displacements (or velocities) in Cartesian space seen from the frame of expression. A Jacobian expressed in frame $\mathcal{F}_i$ (noted for commodity ${}^j\boldsymbol{J}$) can be expressed in frame $\mathcal{F}_j$ thanks to the relation of transformation of the Jacobian matrix

$${}^j\boldsymbol{J} = \begin{bmatrix} {}^j\mathbf{R}_i & \mathbf{0}_3 \\ \mathbf{0}_3 & {}^j\mathbf{R}_i \end{bmatrix} {}^i\boldsymbol{J} \tag{1.16}$$

where ${}^j\boldsymbol{R}_i$ is the 3×3 orientation matrix of frame $\mathcal{F}_i$ expressed in frame $\mathcal{F}_j$.

The geometric Jacobian matrix is also used to define the static force model of robots, used to compute the net joint force vector $\boldsymbol{\tau} = (\tau_1 \ \dots \ \tau_N)^\intercal$ that balances the endpoint P force and moment $-\mathbf{f}_P = (-\mathbf{F}^\intercal \ -\mathbf{M}_P{}^\intercal)^\intercal$ acting on the end effector. To do that, let us define the virtual articular displacements vector $\boldsymbol{\delta}\mathbf{q} = (\delta q_1 \ \dots \ \delta q_N)^\intercal$ and the end effector virtual displacements vector $\boldsymbol{\delta}\mathbf{p}_P = (\boldsymbol{\delta}\mathbf{x_P}^\intercal \ \boldsymbol{\delta}\boldsymbol{\phi}^\intercal)^\intercal$. The *principle of virtual work* states that equilibrium is reached if, and only if, the virtual work done by the forces and moments is zero for arbitrary virtual displacements that conform

to geometric constraints [15].

$$\begin{aligned} 0 &= \tau_1 \delta q_1 + \tau_2 \delta q_2 + \dots \tau_N \delta q_N - \mathbf{F}^\intercal \boldsymbol{\delta} \mathbf{x_P} - \mathbf{M}_P{}^\intercal \boldsymbol{\delta \phi_P} \\ &= \boldsymbol{\tau}^\intercal \boldsymbol{\delta q} - \mathbf{f}_P{}^\intercal \boldsymbol{\delta p} \ . \end{aligned} \tag{1.17}$$

The conformation to geometric constraints is ensured by differential relation Eq. (1.4). Hence, Eq. (1.17) becomes

$$\begin{aligned} 0 &= \boldsymbol{\tau}^\intercal \boldsymbol{\delta q} - \mathbf{f}_P{}^\intercal J \boldsymbol{\delta q} \\ &= (\boldsymbol{\tau} - \boldsymbol{J}^\intercal \mathbf{f}_P)^\intercal \boldsymbol{\delta q} \ . \end{aligned} \tag{1.18}$$

Eq. (1.18) is valid for any virtual arbitrary infinitesimal displacement $\boldsymbol{\delta q} \in \mathbb{R}^N$. Therefore,

$$\boldsymbol{\tau} = \boldsymbol{J}^\intercal \mathbf{f}_P \tag{1.19}$$

1.1.5 Inverse modelling of robotic systems

As was stated in Section 1.1.2, it is more convenient to specify orders in terms of a task. It is thus necessary to be able to translate these commands in terms that the robot may understand. That is the role of inverse modelling. Many methods exist to reverse the direct modelling and some of them will be explained in the following. The focus will be made on redundant manipulators, *i.e.* manipulators that have more degrees of freedom than necessary to perform tasks.

1.2 Redundancy resolution in the literature

1.2.1 Redundancy resolution

This chapter, which is dealing with the notion of kinematics in robotics, explains how these joint and task spaces can be mapped into one another, and specifically

for redundant systems. The relation between the vectors describing the articular configuration and the one describing the task can be specified at the position level, at the velocity (first differential) level and at the acceleration (second differential) level. When a manipulator has more joints than necessary to perform a task, it is said to be redundant. A manipulator is not intrinsically redundant but rather there exist tasks that make a robot redundant. For redundant manipulators, there exists no bijective map between the two spaces, but rather a surjective one from joint space to task space. In fact, for redundant robots, a given regular task can even be performed by an infinite set of solutions in joint space. To perform the task, a choice has then to be made within this set, and this is the purpose of *redundancy resolution*. This choice can be either perceived as an additional complexity to deal with, or an opportunity to enhance the performances of the system in a given context.

While the literature is mainly focused on the velocity level, the redundancy resolution contributions of this doctoral work focus on the position level. Many industrial processes currently performed by robots on shopfloors, especially on aircraft production shopfloors, don't require the robot to move while its tool is operating. Actually, the demand for high precision rarely gives ground to any displacement of the articulated system while the process is being performed. Drilling, screwing or riveting are among these discrete and static operations requiring high precision. This led us to develop tools within this framework (see Section 1.3) of redundancy resolution at the position level. It is useful to consider the direct geometric model (Eq. (1.2)) when using this framework. Regardless, a major part of redundancy resolution schemes presented in the literature focuses on velocity (and acceleration) levels. The common idea of these schemes is to identify and exploit the local behaviour of the system, set in a given articular posture, with regard to the local task displacements, task velocities or accelerations. The mathematical abstraction of this idea is embodied in the Jacobian matrix of the robot task. This central tool is used extensively where redundancy resolution at these levels is concerned. These schemes naturally target industrial applications requiring a continuous motion of the robot while the process is being performed. These applications include for example welding, gasketing, or

camera measurements. It is interesting in this framework to consider the first order differential kinematics model (Eq. (1.3)).

The presentation of some relevant first order differential kinematic schemes is the purpose of this section, while the presentation of the framework for solving redundancy at the position level will be discussed in the next.

1.2.2 Singular Value Decomposition and general solution to inverse kinematics for Redundant manipulators

The analysis of the task Jacobian matrix is of primal importance to understand the motion capabilities of the system. To analyse the linear mapping from joint space to task space, the *Singular Value Decomposition* (SVD) of the Jacobian matrix brings a lot of information. The SVD method was independently discovered by Eugenio Beltrami and Camille Jordan in 1873 and 1874 and the proof of the singular value decomposition for rectangular matrices was published in 1936 in [16]. A reliable and fast algorithm used to compute the SVD of rectangular matrices was introduced in [7]. Eventually, the SVD computation was adapted and made faster for robotic systems Jacobian matrices, that took advantage of their particular kinematic structures in [8].

The SVD decomposes a matrix $\boldsymbol{J_t}$ into the multiplication of 3 matrices U, Σ and $V^\intercal$ as:

$$\boldsymbol{J_t} = \boldsymbol{U\Sigma V^\intercal} \tag{1.20}$$

$$\boldsymbol{J_t} = \underbrace{\begin{pmatrix} \mathbf{u_1} & \dots & \mathbf{u_M} \end{pmatrix}}_{M\times M} \underbrace{\begin{pmatrix} \sigma_1 & & (0) & 0 & \dots & 0 \\ & \ddots & & \vdots & & \vdots \\ (0) & & \sigma_M & 0 & \dots & 0 \end{pmatrix}}_{M\times\big(M+(N-M)\big)} \underbrace{\begin{pmatrix} \mathbf{v_1}^\intercal \\ \vdots \\ \mathbf{v_N}^\intercal \end{pmatrix}}_{N\times N}$$

The columns of $\boldsymbol{U}$ form an orthonormal basis $\mathcal{B}_\mathbf{t} = (\mathbf{u_i})_{i\in[\![1..M]\!]}$ of the task space

displacement/velocity vectors. The columns of $\boldsymbol{V}$ form an orthonormal basis $\mathcal{B}_{\mathbf{q}} = (\mathbf{v_i})_{i\in[\![1..N]\!]}$ of the joint displacement/velocity vectors. $\boldsymbol{\Sigma} = [\boldsymbol{D}\ \mathbf{0}]$ is the $M \times N$ matrix whose $M \times M$ submatrix $\boldsymbol{D}$ is a diagonal matrix containing the singular values σ_i, in decreasing order, of the Jacobian matrix $\boldsymbol{J_t}$ ($\boldsymbol{D} = diag((\sigma_i)_{i\in[\![1..M]\!]})$). We will be focusing on redundant robots, therefore $N > M$. The main interest about these bases is that they are linked together through simple relations involving the Jacobian matrix:

$$\begin{cases} \forall i \in [\![1..M]\!], \boldsymbol{J_t}\mathbf{v_i} = \sigma_i \mathbf{u_i} \\ \forall i \in [\![M+1..N]\!], \boldsymbol{J_t}\mathbf{v_i} = \mathbf{0} \end{cases}. \tag{1.21}$$

(Note: if $\boldsymbol{J_t}$ is of rank $R < M$, $\forall i \in [\![R+1..M]\!], \sigma_i = 0$).

From Eq. (1.20) and Eq. (1.21), one can write that:

$$\boldsymbol{J_t} = \sum_{i=1}^{M} \sigma_i \mathbf{u_i}\mathbf{v_i}^\intercal$$

So, it can be understood that for any articular velocity $\mathbf{V_q}$, the task velocity produced by $\mathbf{V_q}$ is equal to the $\Big(\sigma_i \langle \mathbf{v_i}, \mathbf{V_q} \rangle\Big)$-weighted decomposition of the task singular $\mathbf{u_i}$ vectors, as :

$$\boldsymbol{J_t}\mathbf{V_q} = \left(\sum_{i=1}^{M} \sigma_i \mathbf{u_i}\mathbf{v_i}^\intercal\right)\mathbf{V_q} = \sum_{i=1}^{M} \sigma_i \mathbf{u_i}(\mathbf{v_i}^\intercal \mathbf{V_q}) = \sum_{i=1}^{M} \Big(\sigma_i \langle \mathbf{v_i}, \mathbf{V_q} \rangle\Big)\mathbf{u_i} \tag{1.22}$$

From Eq. (1.21), we can gather that the R first $\mathbf{v_i}$ vectors left-multiplied by $\boldsymbol{J_t}$ span the realisable task velocities space. If $R < M$, tasks that have components in $\text{span}\{\mathbf{u_{R+1}} \dots \mathbf{u_M}\}$ won't be fully realisable from the current configuration, which is said to be *singular*. With the SVD formalism, rank deficiency of the Jacobian matrix naturally translates into a loss of mobility along certain directions of the task velocity space. On the other hand, the $N - R$ last $\mathbf{v_i}$ vectors correspond to articular velocities that produce zero task velocities. These last joint velocity vectors thus span the null

space of the task Jacobian matrix: $\ker \boldsymbol{J_t} = \text{span}\{\mathbf{v_{R+1}}, \dots \mathbf{v_N}\}$.

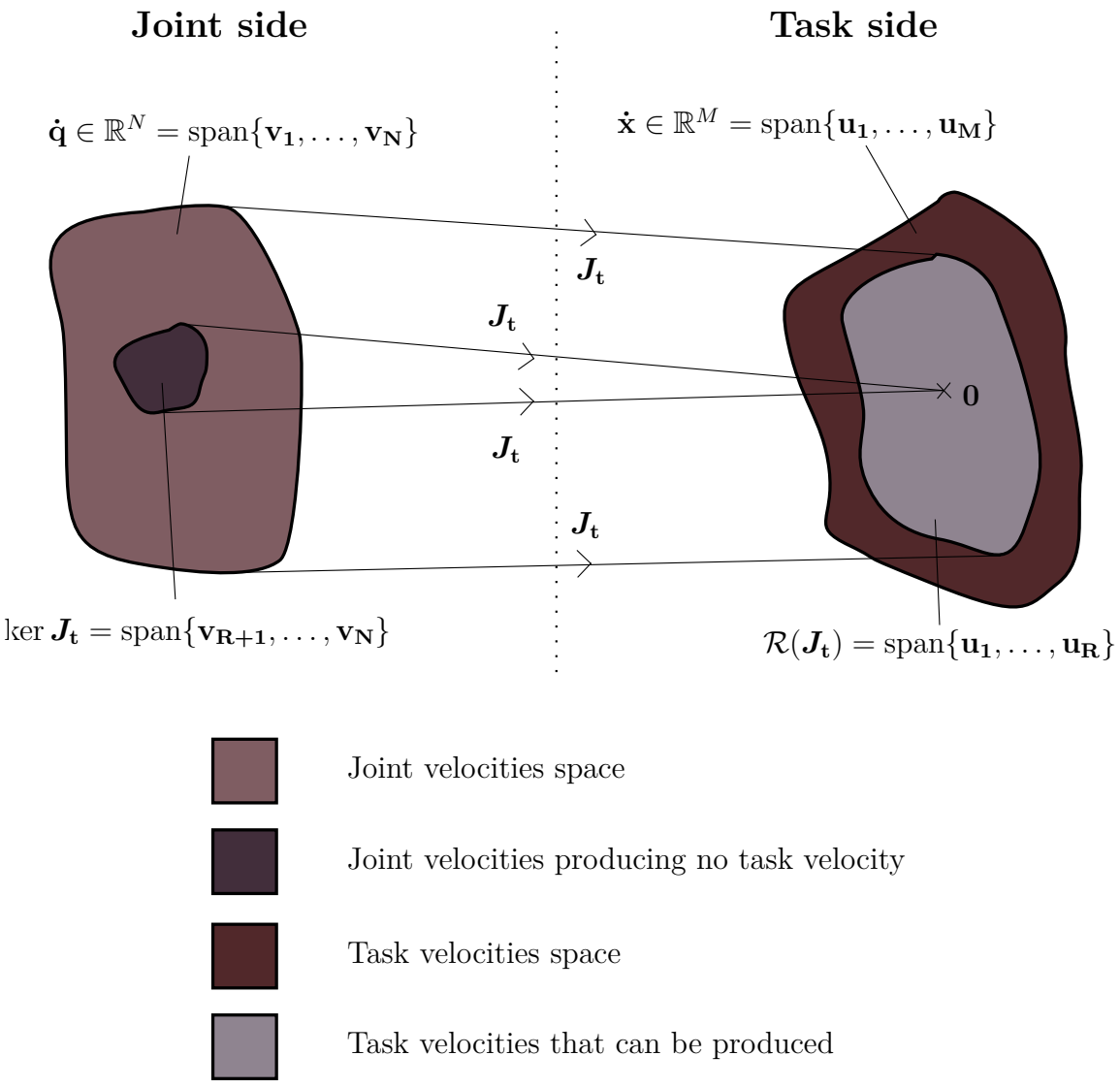


Figure 1.2.1: Task space and joint space representation with SVD formalism.

Of course, the mapping can easily be reversed, as a task velocity along $\mathbf{u_i}, (i \leq R)$ can only be performed if an articular velocity along $\mathbf{v_i}$ is performed on the joints

side.Hence, any task velocity can be written as a linear combination of the base vectors $\mathbf{u_i}$.

$$\begin{gathered}\forall \mathbf{V_t} \in \text{span}\{B_\mathbf{t}\} = \mathbb{R}^M, \\ \exists!(\alpha_1, \ldots, \alpha_M) \in \mathbb{R}^M : \mathbf{V_t} = \alpha_1 \mathbf{u_1} + \cdots + \alpha_M \mathbf{u_M}\end{gathered}$$

Therefore, the same linear combination of the corresponding $\mathbf{v_i}$, weighted by the ratio $1/\sigma_i$ (see Eq. (1.23)), is the minimal norm solution to the inverse kinematics problem. In any configuration $\mathbf{q}$ that is not singular (*i.e.* $\sigma_M(\mathbf{q}) > 0$) :

$$\begin{gathered}\mathbf{V_q} = \frac{\alpha_1}{\sigma_1}\mathbf{v_1} + \cdots + \frac{\alpha_M}{\sigma_M}\mathbf{v_M} \text{ verifies :} \\ \boldsymbol{J_t}\mathbf{V_q} = \mathbf{V_t}\end{gathered} \tag{1.23}$$

Interestingly, any combination of the articular velocities spanned from $\ker \boldsymbol{J_t} = \text{span}\{\mathbf{v_{R+1}}, \ldots \mathbf{v_N}\}$ can be added to $\mathbf{V_q}$ without interfering with the achievement of task velocity $\mathbf{V_t}$:

$$\begin{gathered}\forall(\alpha_M + 1, \ldots, \alpha_N) \in \mathbb{R}^{N-M}, \\ \begin{aligned}\boldsymbol{J_t}\left(\mathbf{V_q} + \sum_{i=M+1}^{N} \alpha_i \mathbf{v_i}\right) &= \boldsymbol{J_t}\mathbf{V_q} + \sum_{i=M+1}^{N} \alpha_i \boldsymbol{J_t}\mathbf{v_i} \\ &= \mathbf{V_t} + \mathbf{0} \\ &= \mathbf{V_t}\end{aligned}\end{gathered} \tag{1.24}$$

Although the SVD is a very interesting tool, such a detailed description of the mapping $\boldsymbol{J_t}$ is not always necessary. One may resort to less costly tools to reverse the mapping $\boldsymbol{J_t}$ of a redundant manipulator. One other way is to perform the generalised inversion of the Jacobian matrix. There exist an infinity of generalised inverses $\boldsymbol{G}$ of $\boldsymbol{J_t}$. For a given generalised inverse $\boldsymbol{G}$, the articular velocity $\dot{\mathbf{q}} = \boldsymbol{G}\dot{\mathbf{t}}$ is a least square solution to the end effector task constraint $\dot{\mathbf{t}} = \boldsymbol{J_t}\dot{\mathbf{q}}$. This means that $\dot{\mathbf{q}} = \boldsymbol{G}\dot{\mathbf{t}}$ minimises $\|\dot{\mathbf{t}} - \boldsymbol{J_t}\dot{\mathbf{q}}\|$. When $\boldsymbol{J_t}$ is full-rank, the generalised inverse verifies $\dot{\mathbf{t}} = \boldsymbol{J_t}\dot{\mathbf{q}} =$

$\boldsymbol{J_t}\boldsymbol{G}\dot{\mathbf{t}}$, and thus $\|\dot{\mathbf{t}} - \boldsymbol{J_t}\dot{\mathbf{q}}\| = 0$ and complies exactly to the task.

A particular generalised inverse of $\boldsymbol{J_t}$ is the pseudoinverse $\boldsymbol{J_t}^\dagger$. This generalised inverse also minimises $\|\dot{\mathbf{q}}\|$ (minimum norm solution). Concretely, this means that when $\boldsymbol{J_t}$ is full rank :

$$\boldsymbol{J_t}^\dagger \dot{\mathbf{t}} = \dot{\mathbf{q}} \tag{1.25}$$

is verified, and the unweighted norm of $\dot{\mathbf{q}}$ is as small as it can be. From a SVD perspective, the pseudoinverse always finds the minimum norm solution $\dot{\mathbf{q}}$ which verifies $\dot{\mathbf{q}} \in \text{span}\{\mathbf{v_1}, \ldots, \mathbf{v_R}\} = \mathbb{R}^N \setminus \ker \boldsymbol{J_t}$

The pseudoinverse $\boldsymbol{A}^\dagger$ of a matrix $\boldsymbol{A}$ must verify the Moore-Penrose conditions [17]:

$$\boldsymbol{A}\boldsymbol{A}^\dagger\boldsymbol{A} = \boldsymbol{A} \tag{1.26}$$
$$\boldsymbol{A}^\dagger\boldsymbol{A}\boldsymbol{A}^\dagger = \boldsymbol{A}^\dagger \tag{1.27}$$
$$(\boldsymbol{A}\boldsymbol{A}^\dagger)^\intercal = \boldsymbol{A}\boldsymbol{A}^\dagger \tag{1.28}$$
$$(\boldsymbol{A}^\dagger\boldsymbol{A})^\intercal = \boldsymbol{A}^\dagger\boldsymbol{A}. \tag{1.29}$$

When the Jacobian matrix is full rank (and low rectangular), the pseudoinverse can be computed as

$$\boldsymbol{J_t}^\dagger = \boldsymbol{J_t}^\intercal(\boldsymbol{J_t}\boldsymbol{J_t}^\intercal)^{-1}.$$

Otherwise, one can use the SVD of $\boldsymbol{J_t}$ to compute its pseudoinverse, as

$$\boldsymbol{J_t}^\dagger = \boldsymbol{V}\boldsymbol{\Sigma}^\dagger\boldsymbol{U}^\intercal = \sum_{i=1}^{R} \frac{\mathbf{v_i}\mathbf{u_i}^\intercal}{\sigma_i}.$$

The general solution to Eq. (1.3) is the composition of the pseudoinverse solution and of a term taken in the null space of $\boldsymbol{J_t}$:

$$\dot{\mathbf{q}} = \boldsymbol{J_t}^\dagger\dot{\mathbf{t}} + (\boldsymbol{I} - \boldsymbol{J_t}^\dagger\boldsymbol{J_t})\dot{\mathbf{q}}_0. \tag{1.30}$$

This solution may remind the reader of Eq. (1.24), where null space joint velocities

are performed by the manipulator without perturbing the task. The term $\dot{\mathbf{q}}_0$ is an arbitrary joint velocity, and $(\boldsymbol{I} - \boldsymbol{J_t}^\dagger \boldsymbol{J_t})\dot{\mathbf{q}}_0$ corresponds to the orthogonal projection of $\dot{\mathbf{q}}_0$ into the null space of $\boldsymbol{J_t}$. $(\boldsymbol{I} - \boldsymbol{J_t}^\dagger \boldsymbol{J_t})$ is an orthogonal projector of the null space of $\boldsymbol{J_t}$, as can be seen in

$$\boldsymbol{J_t}(\boldsymbol{I} - \boldsymbol{J_t}^\dagger \boldsymbol{J_t}) = \boldsymbol{J_t} - \boldsymbol{J_t}\boldsymbol{J_t}^\dagger \boldsymbol{J_t} = \mathbf{0} \tag{1.31}$$

which is direct a result of Eq. (1.26).

1.2.3 Local optimisation-based velocity level redundancy resolution schemes

Basic principle : In the local optimisation approach, the term $(\boldsymbol{I} - \boldsymbol{J_t}^\dagger \boldsymbol{J_t})\dot{\mathbf{q}}_0$ of Eq. (1.30) is used to stir the articular configuration of the manipulator towards an extremum of an objective real-valued function which depends on the current configuration $H(\mathbf{q})$, without hampering the tracking of the primary task velocity $\dot{\mathbf{t}}$. In this context, H is often called a performance criteria. To maximise (respectively minimise) H, $\dot{\mathbf{q}}_\mathbf{0}$ has to be oriented along the gradient (respectively the anti-gradient) of function $H(\mathbf{q})$ as so (for the maximisation of H) :

$$\dot{\mathbf{q}}_\mathbf{0} = k_H \nabla H(\mathbf{q}) \tag{1.32}$$

with $k_H \geq 0$. Of course, the gradient is a local notion, so it is important not to take a step in the gradient direction that would be too big and lead to a decrease of the performance criteria. From a given configuration $\mathbf{q}$, applying a velocity $\dot{\mathbf{q}}_\mathbf{0}$ - choosing a small enough value for k_H - would definitely stir the articular configuration of the robot in a direction that would improve the performance criteria. The velocity resulting from the projection of $\dot{\mathbf{q}}_\mathbf{0}$ in the null space of the primary task with the $(\boldsymbol{I} - \boldsymbol{J_t}^\dagger \boldsymbol{J_t})$ operator is bound to have at least no hampering effect on the performance criteria, and at best is bound to improve it.

The performance criteria can take various forms. It may be designed to keep away

from kinematic singularities, to avoid obstacles [18] or to stay as far as possible from joints position limits [18, 19]. An overview and a discussion of some well-known criteria (mostly related to kinematic considerations) can be found in [20].

singularity and workspace ill-conditioning avoidance : Several criteria were designed to indicate distances from singularities or ill-conditioning of the workspace. The three main ones are the *manipulability measure* μ, the *condition number* κ, and the *smallest singular value* σ_{min}. Using these criteria may improve the matter of avoiding singular configurations, but does not ensure singularity avoidance. Some singularities are unavoidable (wrist and shoulder singularities), and the approach is local and thus can't ensure singularity avoidance on an entire trajectory.

Singular configurations occur where the Jacobian matrix loses rank. In other words, for non redundant manipulators ($M = N$) the occurrence of a singularity can be detected by computing the determinant of the geometric Jacobian matrix. For redundant manipulators, a generalisation of this concept is to compute the determinant of the geometric Jacobian matrix multiplied by its transpose. The manipulability measure is similar to that, as :

$$\mu = \sqrt{|\boldsymbol{J}\boldsymbol{J}^\intercal|}. \tag{1.33}$$

Remarkably, μ is also equal to the product of the singular values of the geometric Jacobian matrix, although their calculation is more costly. A loss of rank of the Jacobian matrix translates into the decrease to zero of the smallest formerly non-zero singular value.

$$\mu = \prod_{i=1}^{M} \sigma_i \tag{1.34}$$

The computation of the manipulability measure using Eq. (1.33) is simple, which makes it an appealling criterion. Some authors say that the manipulability measure defines the degree of conditioning of the workspace [21]. Others [22–24] argue that the manipulability measure may not be a very conclusive indicator of closeness to singularity or ill-conditioning of the workspace. Eq. (1.34) may help see that this criteria does not always vary dramatically near singularities, as the highest singular

values may compensate for the decrease of the smallest ones.

A good measure of the conditioning of the workspace is the aptly named condition number κ. This measure is equal to the ratio between the highest singular value and the smallest one.

$$\kappa = \frac{\sigma_1}{\sigma_M} \tag{1.35}$$

In the SVD, singular values are indicators of the efficiency of the manipulator in a configuration to produce velocities in M orthogonal directions. As such, the highest (and first) singular value indicates the efficiency at producing an operational velocity for the end effector, in the direction in which it is most easily moved. Inversely, the smallest (and last) singular value indicates the efficiency at producing an operational velocity for the end effector, in the direction in which it is most tediously moved. A condition number of 1 implies that all the singular values are equal to 1, and thus it indicates the isotropy of mobility of the manipulator for its end effector in a given configuration. Oppositely, if the position number has a high value, the anisotropy of mobility is also high, and some operational directions will be more easily moved in than others. An alternative to the condition number is its inverse : *the local conditioning index* [20], which is equal to $1/\kappa$. It is often seen as more convenient as its variations are constrained between 0 in singular configurations to 1 in isotropic configurations. These criterion are said to be good measures of a manipulator's distance to singularity [23, 25].

Despite this, measuring closeness to singularity using two singular values may not yield much better results than doing it using all singular values. The variations of one may compensate for another's, and that would lead to an unwavering condition number while the closeness to a singularity changes. Quite remarkably, the simpler criterion of smallest singular value σ_{min} is a good indicator of the closeness to a singular configuration [24].

Collision avoidance : Occurrences of collisions of the links of a redundant manipulator with its environment can be avoided by designing performance criteria based on the minimum distance from collision. In [18], the robot control is made with the

use of potential fields moving the links of the robot away from the obstacles.

joints position limit avoidance : [19] suggests to use optimisation function:

$$H(\mathbf{q}) = \sum_{i=1}^{N} \left(\frac{q_i - q_{i,mid}}{q_{i,max} - q_{i,min}} \right)^2 \tag{1.36}$$

to stir the joint configuration towards the center of the joints range. $q_{i,max}$ and $q_{i,min}$ are respectively the maximum and minimum limits of joint i, $q_{i,mid} = \frac{q_{i,max}+q_{i,min}}{2}$, and q_i denotes the position on joint i.

1.2.4 Augmented Jacobian

Another way of exploiting redundancy is to create a task vector whose dimension M matches the number of degrees of freedom N of the robot. For example, one can append the usual end effector Cartesian trajectory tracking task with other constraint task vectors, until the dimension of the task is N. This provides with a square Jacobian matrix, which can then be inverted without ambiguity. The concept was introduced in [26, 27] and referred to as task-augmentation [28].

Let $\mathbf{t_{A_i}}(\mathbf{q})$ be the i$^{\text{th}}$ task vector. Let $\mathbf{t_A}$ be the augmented task vector constructed from the concatenation of all the sub-tasks $\mathbf{t_{A_i}}, i \in [\![1..P]\!]$.

$$\mathbf{t_A} = \begin{bmatrix} \mathbf{t_{A_1}} \\ \vdots \\ \mathbf{t_{A_p}} \end{bmatrix} \tag{1.37}$$

The sub-tasks are chosen so that

$$\sum_{i=1}^{p} dim(\mathbf{t_{A_i}}) = N. \tag{1.38}$$

We can define the first order differential equation for each of these tasks:

$$\forall i \in [\![1..p]\!], \dot{\mathbf{t}}_{A_i} = \boldsymbol{J}_{\mathbf{t}_{A_i}} \dot{\mathbf{q}}. \tag{1.39}$$

And eventually define the square augmented matrix $\boldsymbol{J}_A$ as

$$\boldsymbol{J}_A = \begin{bmatrix} \boldsymbol{J}_{\mathbf{t}_{A_1}} \\ \vdots \\ \boldsymbol{J}_{\mathbf{t}_{A_p}} \end{bmatrix}. \tag{1.40}$$

Although the task augmentation approach seems appealing, it suffers from a major flaw: the *algorithmic singularities* [29]. Algorithmic singularities occur when the augmented Jacobian looses rank even though none of the concatenated task Jacobian matrix have lost rank. These algorithmic failures occur when tasks conflict between each other. Concretely, if one of the task is an end effector trajectory, and the second task is related to obstacle avoidance, an algorithmic singularity will occur whenever the trajectory passes through or close to an obstacle. In the vicinity of the obstacle, the end effector trajectory task will want to push the end effector toward the obstacle while the obstacle avoidance task will require the opposite.

A mathematical way of explaining this phenomenon is through the SVD of each task Jacobian matrix. Let the SVD of the Jacobian matrix associated to task $i \in [\![1..p]\!]$ be:

$$\boldsymbol{J}_{\mathbf{t}_{A_i}} = \boldsymbol{U}_{\mathbf{t}_{A_i}} \boldsymbol{\Sigma}_{\mathbf{t}_{A_i}} \boldsymbol{V}_{\mathbf{t}_{A_i}}{}^{\mathsf{T}}. \tag{1.41}$$

Let $(\mathbf{v}_{\mathbf{t_{A_i},j}})_{j \in [\![1..N]\!]}$ be the column vectors of $\boldsymbol{V}_{\mathbf{t}_{A_i}}$. If task i is of dimension $M_i (< N)$, the manipulator will have to use joint velocities taken in $\text{span}\{\mathbf{v}_{\mathbf{t_{A_i},1}}, \ldots, \mathbf{v}_{\mathbf{t_{A_i},M_i}}\}$ to perform it.

Now, an algorithmic singularity will happen if, for two tasks $\mathbf{t}_{\mathbf{A_k}}$ and $\mathbf{t}_{\mathbf{A_l}}$ for which the articular configuration is not singular,

$$\text{span}\{\mathbf{v}_{\mathbf{t_{A_k},1}}, \ldots, \mathbf{v}_{\mathbf{t_{A_k},M_k}}\} \cap \text{span}\{\mathbf{v}_{\mathbf{t_{A_l},1}}, \ldots, \mathbf{v}_{\mathbf{t_{A_l},M_l}}\} \neq \varnothing. \tag{1.42}$$

Indeed, this non empty intersection implies that performing one task will always interfere with the other task, thus leading to an incompatibility between the two tasks. A singularity will occur, while none of the sub-task Jacobian matrices have lost rank. In fact, the only way not to have an algorithmic singularity is when all the singular joint velocity vectors of all tasks are orthogonal to each others.

1.2.5 Task priority

Instead of trying to fulfill all the tasks together, the task priority strategy [30–33] uses a prioritisation of each task to cope with the algorithmic singularities problem. In this framework, an order of priority is defined between all the tasks, and the lower priority tasks only produce a null space motion which does not interfere with the higher priority tasks. As such, lesser tasks may not always be fulfilled completely as the joint velocity associated to them are projected onto the null space of more important tasks.

The joint velocity $\dot{\mathbf{q}}^{[1]}$ associated to the most important (the first) task is computed first using the pseudoinverse of the associated Jacobian matrix. Then, a recursive operation is performed for each lesser task in order of decreasing priority. The joint velocity $\dot{\mathbf{q}}^{[k]}$ that is outputed at iteration k is the addition of

- the joint velocity $\dot{\mathbf{q}}^{[k-1]}$ of the previous iteration
- with the projection in the null space of all the higher priority tasks of
  - the joint velocity required to comply with task k
  - minus the error made on task k by applying joint velocity $\dot{\mathbf{q}}^{[k-1]}$

$$\dot{\mathbf{q}}^{[1]} = \boldsymbol{J}^{\dagger}_{\mathbf{t}_{A_1}} \dot{\mathbf{t}}_{A_1}$$

$$\dot{\mathbf{q}}^{[k]} = \underbrace{\dot{\mathbf{q}}^{[k-1]}}_{④} + \underbrace{\left(\boldsymbol{I} - \begin{bmatrix} \boldsymbol{J}_{\mathbf{t}_{A_1}} \\ \vdots \\ \boldsymbol{J}_{\mathbf{t}_{A_{k-1}}} \end{bmatrix}^{\dagger} \begin{bmatrix} \boldsymbol{J}_{\mathbf{t}_{A_1}} \\ \vdots \\ \boldsymbol{J}_{\mathbf{t}_{A_{k-1}}} \end{bmatrix} \right)}_{③} \underbrace{\boldsymbol{J}^{\dagger}_{\mathbf{t}_{A_k}} (\dot{\mathbf{t}}_{A_k}}_{①} - \underbrace{\boldsymbol{J}_{\mathbf{t}_{A_k}} \dot{\mathbf{q}}^{[k-1]}}_{②})$$

(1.43)

At step k, $\dot{\mathbf{q}}^{[k]}$ must:

① Comply with task velocity $\dot{\mathbf{t}}_{A_k}$

② ...which was modified by the joint velocity vector computed for the $k-1$ first tasks (modifications of task k velocity : $\boldsymbol{J}_{\mathbf{t}_{A_k}} \dot{\mathbf{q}}^{[k-1]}$),

③ ...all that without interfering with the $k-1$ first tasks (null space projection)

④ ...while complying as best as possible with the $k-1$ first tasks (result of the previous operation).

When the manipulator is out of the algorithmic singularities that could be spotted using the task augmentation implementation, both the task priority and the task augmentation methods provide the same result. The advantage of this prioritisation strategy is that even if an algorithmic singularity occurs, the primary task is always fulfilled (as long as its associated Jacobian remains full rank). However, when approaching an algorithmic singularity, the secondary term {③, ①, ②} may produce very high joint velocities, which may not be feasible. To cope with this problem, a strategy consists in removing the correctional term $\boldsymbol{J}_{\mathbf{t}_{A_k}} \dot{\mathbf{q}}^{[k-1]}$ in the formula of Eq. (1.43). Task k is then less efficiently tracked, but at least the joint velocities remain reasonable near algorithmic singularities. The recursive formula then becomes :

$$\dot{\mathbf{q}}^{[1]} = \boldsymbol{J}^{\dagger}_{\mathbf{t}_{A_1}} \dot{\mathbf{t}}_{A_1}$$

$$\dot{\mathbf{q}}^{[k]} = \dot{\mathbf{q}}^{[k-1]} + \left(\boldsymbol{I} - \begin{bmatrix} \boldsymbol{J}_{\mathbf{t}_{A_1}} \\ \vdots \\ \boldsymbol{J}_{\mathbf{t}_{A_{k-1}}} \end{bmatrix}^{\dagger} \begin{bmatrix} \boldsymbol{J}_{\mathbf{t}_{A_1}} \\ \vdots \\ \boldsymbol{J}_{\mathbf{t}_{A_{k-1}}} \end{bmatrix} \right) \boldsymbol{J}^{\dagger}_{\mathbf{t}_{A_k}} (\dot{\mathbf{t}}_{A_k}) \tag{1.44}$$

When using these strategies, a positional drift will occur for all tasks that could not be fulfilled completely for the sake of not tempering with the higher ranking ones. As they are, the strategies blindly look at task velocities and do not take into account or correct any task position error. To do so, one can resort to a CLIK implementation by swapping, in Eq. (1.43) or Eq. (1.44), term $\dot{\mathbf{t}}_{A_k}$ with the term $\dot{\mathbf{t}}_{A_k} + \boldsymbol{K}_{A_k}(\mathbf{t}_{A_k} - \mathbf{f}_{\mathbf{t}_{A_k}}(\mathbf{q}))$. This term also integrates a weighted task position error vector, creating a proportional + feedforward control scheme. $\boldsymbol{K}_{A_k}$ is generally a diagonal weight matrix, $\mathbf{t}_{A_k}$ is the desired task position and $\mathbf{f}_{\mathbf{t}_{A_k}}(\mathbf{q})$ is the actual position with regard to task k (direct geometry mapping).

Sometimes, a strict task priority ordering is not easy to decide. In dynamic and changing environments, the priorities may change, or may become antagonistic. Strict task priority strategy may not be absolutely suitable to this context. In [34] is described a soft hierarchy scheme making use of a generalized task weighting strategy that ensures smooth transitions between tasks.

Another strategy called Generalized Hierarchical Control was developed in [35] to allow mixing strict task hierarchies with non-strict ones. Strict task priority strategies are typically based on the concept of task projectors, as described above, while non strict task priority ones rely on a weighting strategy between tasks. The GHC allows to seamlessly transition from one -strict and non strict- tasks hierarchy ordering to another by using a generalised projector which ensures task priorities, transitions, insertions and deletions. A priority matrix is modulated to change the task priorities without modifying the control problem formulation.

1.2.6 Saturation in Null Space (SNS)

While all these schemes, ranging from the local optimisation scheme (Section 1.2.3) to the task prioritisation (Section 1.2.5) one, may integrate a notion of soft joint mechanical limit avoidance, none will ensure that these limits are not reached if they are not defined as primary tasks. Yet, these joint limits (at the position, velocity or acceleration level) are a sine qua non condition to the feasibility of any trajectory. Violating any of these hard constraints is conceptually impossible and thus would prevent any other tasks from being performed. In that regard, these bounds have to be considered as another paradigm. Such a strategy was presented by Omrčen in [36], later revisited by Flacco et al. in [37] as the concept of *saturation in the null space*, which was also adapted to the task prioritisation scheme in [38].

When the pseudoinverse solution to the inverse kinematics problem is found to overdrive some joints hard bounds, the method proceeds by applying the hard bound level to the most overdriven joint and then by redistributing the excess of task velocity/acceleration within the joints that are not yet saturated. The method is applied recursively to the most overdriven joints until too few joints remain to perform the M-dimensional task. In that case, the SNS method applies a subunitary scaling factor to the task. This scaling factor eases the task tracking while retaining its geometrical path. One significant advantage of SNS strategies is that singularities are naturally handled either by mere avoidance, as their occurrence along the way would provoke excess of joint speed or acceleration, or simply by slowing down the overall motion, by decreasing the value of the scaling factor.

1.3 Redundancy spaces - A position-level approach to solving redundancy

1.3.1 Redundancy space formulation

Context : In typical aircraft assembly processes, robots are used to perform high quality, high precision operations. Aircraft manufacturers increasing expectations

now regularly set mechanical tolerances below 0.2 mm. Typical industrial robot performances are within a repeatability[3] range $[0.05mm, 0.5mm]$, and within an accuracy[4] range $[1mm, 5mm]$. Consequently, high precision processes usually happen after the robot has accurately positioned itself statically, using local calibration techniques. After accurately positioning itself, the robot is generally asked to remain static while its tool performs the process (be it drilling, riveting, screwing, or else). During the interaction, the task performed by the robot is seemingly the simplest one : to keep a static articular configuration until the operation is finished. In this context, the system can naturally be called a positioner. The goal for the robot arm is to provide an unperturbed moment for the tool to perform its process. No clever motion control is possible during the process, because any geometric deviation at the process interface may ruin the quality of the operation result, or cause breakage. Beside the Aircraft industry, a lot of robotised operations encountered in the manufacturing industry are defined as *positional* tasks (to be opposed to trajectory tasks/tasks were the manipulator is moving). Pick and place operations define grasping and releasing location for the gripper. Assembly operations such as riveting or screwing also define end effector positional locations.

To perform these positional tasks, our use case involves a system composed of a mobile platform (KMP) and a 7-DOFs serial arm (LBR-iiwa). Together, these robots accumulate 10 DOFs, which makes the full system extremely redundant with regards to many types of end-effector positional tasks.

Objectives : Redundancy is sometimes perceived as a source of complexity. But while it adds complexity, redundancy also adds possibilities. It provides a choice in the articular configuration to use. Therefore, redundancy requires taking decisions that were not made for non-redundant systems. We want to make the possibilities redundancy offers at the position level clearly identifiable to robot end-users. We

[3]Repeatability is a measure of the reproducibility of an end effector pose for a specified articular position.

[4]Accuracy is defined as the misplacement between a desired end effector pose and the average of the end effector poses produced by the robot.

identified a clear need for a simple formulation of redundancy at the position level, that allows to know the extent of solutions redundancy offers. Additionally, any of these possibilities must be computable quickly and without approximation.

Expressing positional redundancy for positional tasks using differential schemes : All the schemes described in Section 1.2 solve redundancy at the velocity or acceleration level. Getting back to the position level requires the time integration of the solutions. If a task $\mathbf{t}$ is to hold the end effector static, a redundant manipulator can theoretically generate articular velocities that won't disturb the task. The task velocity $\dot{\mathbf{t}}$ is in that case zero at all time, and therefore, for $\dot{\mathbf{q}}_0 \in \mathbb{R}^N$:

$$\begin{aligned} \dot{\mathbf{q}} =& \boldsymbol{J_t}^{\dagger}\dot{\mathbf{t}} + (\boldsymbol{I} - \boldsymbol{J_t}^{\dagger}\boldsymbol{J_t})\dot{\mathbf{q}}_0 && (1.45) \\ =& (\boldsymbol{I} - \boldsymbol{J_t}^{\dagger}\boldsymbol{J_t})\dot{\mathbf{q}}_0, && (1.46) \end{aligned}$$

will generate a self-motion of the manipulator with regards to the task. Integrating this velocity over time should provide with a set of articular solutions to the problem. Equivalently, given the Taylor expansion developed in Eq. (1.4), the variation of task $\boldsymbol{\delta}\mathbf{t}$ caused by an articular variation $\boldsymbol{\delta}\mathbf{q}$ is given by :

$$\boldsymbol{\delta}\mathbf{t} = \mathbf{F_t}(\mathbf{q} + \boldsymbol{\delta}\mathbf{q}) - \mathbf{F_t}(\mathbf{q}) = \boldsymbol{J_t}\boldsymbol{\delta}\mathbf{q} + \begin{bmatrix} \mathcal{O}(\|\boldsymbol{\delta}\mathbf{q}\|^2) & \dots & \mathcal{O}(\|\boldsymbol{\delta}\mathbf{q}\|^2) \end{bmatrix}^{\mathsf{T}}. \tag{1.47}$$

If the articular variation is chosen to be in the null-space of the Jacobian matrix, *i.e.* $\boldsymbol{\delta}\mathbf{q} \in \ker \boldsymbol{J_t}$, the task positional variation becomes

$$\boldsymbol{\delta}\mathbf{t} = \begin{bmatrix} \mathcal{O}(\|\boldsymbol{\delta}\mathbf{q}\|^2) \\ \vdots \\ \mathcal{O}(\|\boldsymbol{\delta}\mathbf{q}\|^2) \end{bmatrix},$$

which is close to nothing if $\boldsymbol{\delta}\mathbf{q}$ is small.

Numerical bias : Repeating one of these scheme several times will therefore provide with configurations that comply with the task $\mathbf{t}$ with a task precision which is given by the remaining terms of the Taylor expansion. To improve this matter, more terms

of the Taylor expansion could be taken into account in the inverse kinematic step. However, no scheme that uses this strategy or its improvement will ever provide accurate results efficiently. The solution found by doing these repeated computations suffer from obvious numerical biases leading to problems of task imprecision.

Incomplete exploration of redundancy : Another important point, beside the inaccuracy of the solution based on a velocity-level approach, is the fact that the possibilities offered by positional redundancy can hardly be all explored. In the scheme described above, to choose $\boldsymbol{\delta q} \in \ker \boldsymbol{J_t}$, one solution is to generate an arbitrary $\dot{\mathbf{q}}_0 \in \mathbb{R}^N$ and to project it in the null space of the Jacobian matrix thanks to the projector $(\boldsymbol{I} - \boldsymbol{J_t}^\dagger \boldsymbol{J_t})$. However, choosing arbitrarily $\dot{\mathbf{q}}_0$ will not provide an efficient way of exploring the entire space of solutions, especially if the redundancy of the system is of order $s > 1$. At most one discretely described curve of configurations complying with the task will be found. Another, more elegant, yet more expensive solution, could be to compute the SVD of the Jacobian matrix in each new sampled task-complying configuration. The eigenvectors spanning the null space of the Jacobian matrix, *i.e.* $\{\mathbf{v_{R+1}}, \ldots, \mathbf{v_N}\}$, are independent articular velocities that generate no task velocity. Therefore, taking small steps in the articular directions suggested by each of these vectors could help explore more thoroughly the postural possibilities redundancy offer thanks to an incremental grid exploration approach. However, a common drawback of solutions based on differential kinematic is the lack of *cyclicity* [28]. This property implies that no representative sampling of the possibilities offered by redundancy can be easily computed using differential kinematic schemes.

No positional redundancy boundaries : Another drawback of differential schemes, which is also related to cyclicity, is the fact that boundaries of the solution space cannot be easily formulated or found. This is a very important feature to be able explore its the full extent.

Sources of positional redundancy : In our study, redundancy is defined for a manipulator and a positional task. It therefore emanates from two tangled sources. Additional motion capabilities (extra actuators) is one, removal of constraints on the

task is the other. A n-DOFs manipulator used for a positional task $\mathbf{t}$ of dimension $m < n$ is said to be kinematically redundant, with a redundancy of order $s = n - m$. The dimension of the Jacobian null space will be equal to s in non-singular configurations. This means that in a non-singular configuration, there exist a base of s independent articular displacements (the null space articular velocity vectors that can be issued from a SVD study) which will not produce any displacement of $\mathbf{t}$ for this manipulator[5]. Any combination of these s independent articular displacement vectors will produce no displacement of the task.

Redundancy spaces : In this thesis, we choose to tackle redundancy exploitation with a different approach. An intuition is that for positional tasks, redundancy exploitation based on a velocity or acceleration level formulation of the task is not adapted. Instead, it seems more appropriate to remain at the position level.

A *redundancy space*, i.e. a space parameterised by well chosen, higher level independent parameters, is used to represent the full extent of the solution space of the robot performing its positional task. Let $\boldsymbol{\alpha} = \begin{bmatrix} \alpha_1 & \dots & \alpha_s \end{bmatrix}^T$ be a s-dimensional redundancy space position of the manipulator, each α_i being a redundancy parameter. Setting the redundancy space position allows for an unambiguous input formulation of the inverse geometry problem by reducing the number of unknowns of this problem to m in a system of m equations, and provides a finite set of solutions to the inverse geometry problem of the redundant manipulator. A set of articular solutions may be rigorously computed thanks to an *admissible redundancy space position-admissible task* input couple $(\boldsymbol{\alpha}, \mathbf{t})$. Changing the position in redundancy space (*i.e.* the value of all these parameters) will change the articular configuration of the system without modifying the end-effector positional behaviour. Robots direct geometric models are generally non linear mappings, and reversing them analytically can be tedious, even for non redundant robots. However, for some types of redundant systems, closed-form solutions exist for the inverse geometry. These analytic expressions are very desirable for redundancy space formulations.

[5] in a first order approximation

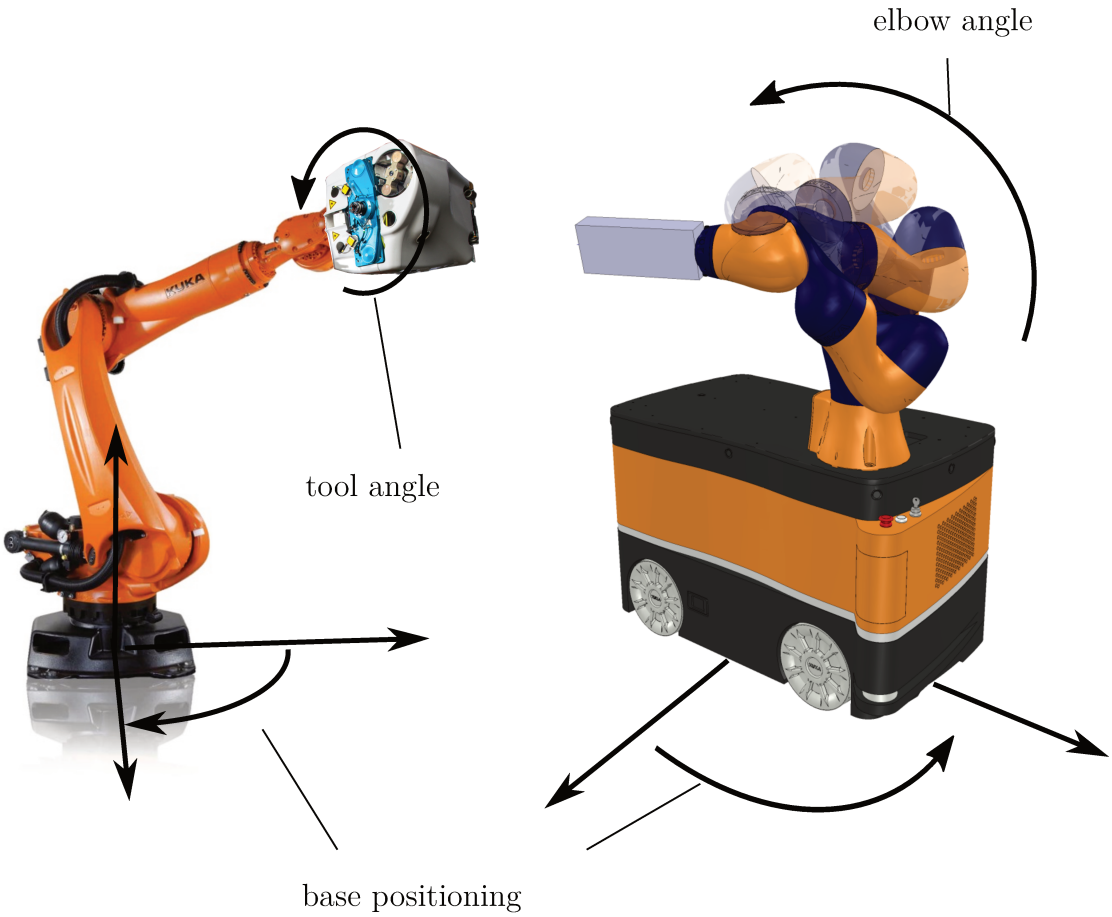


Figure 1.3.1: The parameterisation of positional redundancy for two different systems, for two different tasks. On the left, the task involves directing an axis that is attached to the end effector (drilling operations require just that), which frees up a rotation around this axis. On the right, a classic pose task for the end effector is chosen. The base positioning of both robot arms has degrees of freedom in these example.

Without being a rule of thumb, the parameterisation of the redundancy space can often take inspiration from which geometric constraints were freed up on a fully constrained (pose) task, and which internal/external motion capability were added by "extra" joints in a kinematic chain. There are generally more than a single redundancy space formulation for a system and tasks. A necessary property of this parameterised redundancy space is that picking an *admissible redundancy space position-admissible task* couple leads to a unique set of $k \in \mathbb{N}^{+*}$ articular solutions[6] (when out of singularities). In other words, an inverse geometry step is required to get the lower level inputs that are the joint positions of the manipulator corresponding to the task and position in redundancy space.

Advantages of redundancy space formulation : The advantage of using a redundancy parameterisation in our context is several-fold. For one, having an unambiguous parameterisation of the space of solutions, that can be easily be intuited and understood, is paramount. It helps demystifying the complex notion that is redundancy, and eases communication on the subject. It also exposes the full extent of the solution space of a robot, which may help choosing a suitable configuration, as will be done in the following chapters. It may help verify quickly and without a doubt a reachability information, or the existence of an inverse geometric solution. Additionally, having an analytic expression of the inverse geometry ensures that no approximation is made on the solution.

1.3.2 An intrinsic redundancy of SRS robots

A first redundancy related to the use of Spherical-Rotary-Spherical (SRS) robots (which are a type of 7-DOFs robots), for the task of positioning (3 geometric constraints) and orientating (3 geometric contraints) the end-effector, is the elbow angle redundancy. We will begin the formulation of a redundancy space for SRS robots by expressing a closed-form solution to this inverse geometry problem. The positional

[6]The existence of this set of k solutions comes from the periodicity of the trigonometric functions used in the geometric model and is unrelated to kinematic redundancy, as it does not increase the dimension of the redundancy space, but rather multiplies the number of solutions by k.

task at hand is a fully constraining positional task.

1.3.2.1 Direct geometric modelling of a iiwa-type robot

For the system we study and use throughout this thesis, the relevant parameters and vectors used to build its DH description are displayed on Fig. 1.3.2, where the LBR iiwa is set in its so-called *zero position*. Joint angles $(q_1, \ldots, q_7)$ are displayed aside their corresponding joint. All the joints of this robot are revolute. $(\mathbf{z}_0, \ldots, \mathbf{z}_7)$ are the $\mathbf{z}$ axes of the links frames, and are situated upon the rotation axes of the robot joints. The $\mathbf{x}$ axes of all these frames are normal to the plan of the picture, point to the back of the picture (are directed like the gaze of the reader), and aren't displayed on it. Relevant distances for the DH parameterisation l_{bs}, l_{se}, l_{ew} and l_{wt}, are the base-to-shoulder, shoulder-to-elbow, elbow-to-wrist and wrist-to-tip distances. They are displayed on the right hand side of the figure.

Table 1.3.1: Geometric parameters of the DH parameterisation of the 7 DOFs LBR iiwa

i	α_i	a_i	d_i	q_i
1	0	0	l_{bs}	q_1
2	$-\pi/2$	0	0	q_2
3	$\pi/2$	0	l_{se}	q_3
4	$\pi/2$	0	0	q_4
5	$-\pi/2$	0	l_{ew}	q_5
6	$-\pi/2$	0	0	q_6
7	$\pi/2$	0	l_{wt}	q_7

The modified DH parameterisation is synthesised in Table 1.3.1 and explained hereinafter. Joint 1 has a vertical orientation and its origin has an offset of l_{bs} from the robot base reference frame origin O, along joint 1 axis of rotation. Joint 2 is perpendicular to joint 1. Their axes of rotation intersect in their shared origin, which is called the *shoulder point* S. Joint 3 is perpendicular to joint 2 and has an offset of l_{se}, along its axis of rotation, from S. Hence, its axis of rotation intersects its two preceding joints' in S. Joint 4 is perpendicular to joint 3 and their axis of rotation

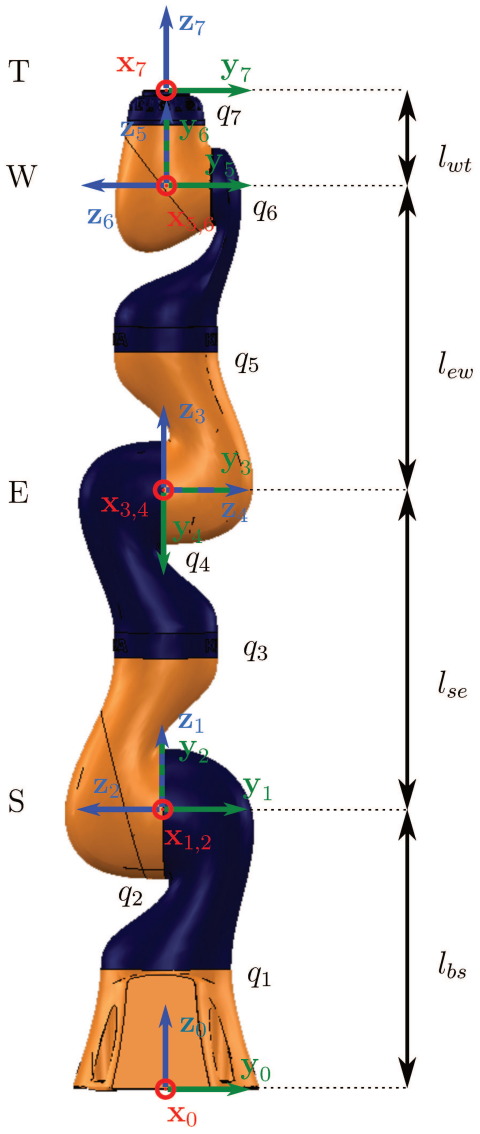


Figure 1.3.2: LBR iiwa in zero position.

intersect it their shared origin, which is called the *elbow point* E. Joint 5 is perpendicular to joint 4 and has an offset of l_{ew}, along its axis of rotation, from E. Joint 6 is perpendicular to joint 5 and their axis of rotation intersect in their shared origin, which is called the *wrist point* W. Joint 7 is perpendicular to joint 6 and has an offset of l_{wt}, along its axis of rotation, from W.

1.3.2.2 Inverse geometric model of a iiwa-type robot

The LBR iiwa is a serial, seven revolute joints robotic arm. Its kinematic structure is said to be anthropomorphic in that it resembles the one of a human arm. This kinematic structure is very popular in robotics because it is known to have very good characteristics in terms of dexterity.

This claim can be correlated to the fact that human beings are among the most nimble beings of animal kingdom. On a more practical ground, the robots presented on Fig. 1.3.3 and the LBR iiwa all present a closed-form inverse geometric solution for the 6-DOFs task of positioning and orientating their extremal link. Closed-form solutions are always to be sought because they present good computational characteristics, including accuracy and efficiency. The following will present how this analytical solution can be derived for the LBR iiwa System.

1.3.2.2.1 LBR iiwa geometric facts and elbow angle

An geometric analysis allow us to simplify the serial structure of the LBR iiwa, as can be seen in Fig. 1.3.4 and Fig. 1.3.5. By construction, its first three joints axes always intersect in the shoulder point S and the three last joint intersect in the wrist point W. Either group of these three intersecting serially attached revolute joints form a equivalent spherical joint respectively centered in S and W. The fourth joint is centered in the elbow point E. This decomposition of the structure of the arm gives the name of Spherical-Rotary-Spherical systems.

As can be seen in Fig. 1.3.4, the pose location of the robot base reference frame

Figure 1.3.3: Other examples of 7 DOFs anthropomorphic arms, and SRS type robots. Collaborative robots Franka Emika Panda, KUKA LWR4 and ABB Yumi.

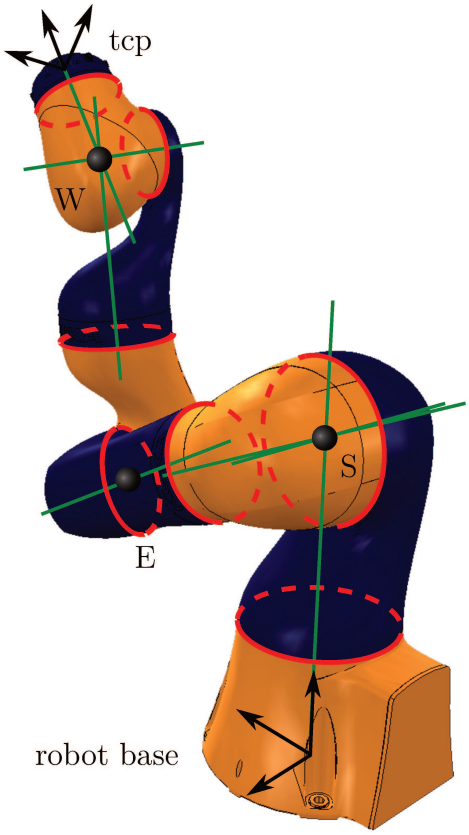


Figure 1.3.4: The Spherical Rotary Spherical (S-R-S) structure of the LBR iiwa.

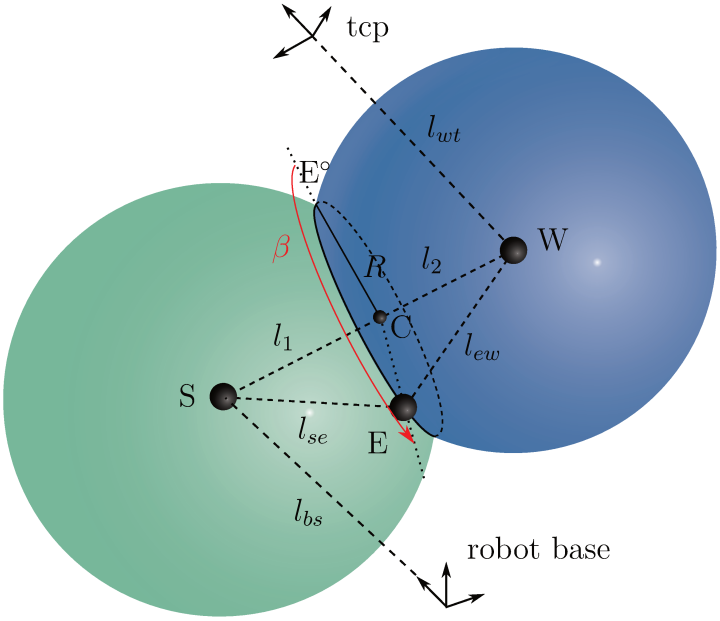


Figure 1.3.5: LBR iiwa geometric problem simplification and elbow angle definition.

fully constrains the position of the shoulder point S. Therefore, attaching the robot to a fixed location imposes the position of the shoulder S. Similarly, the desired pose location of the tool center point (TCP) fully constrains the position of the wrist point W. The upper arm and forearm of the robot have fixed lengths, therefore the distances from S to E (l_{se}) and from E to W (l_{ew}) have constant values. The maximum distance separating S to W is therefore $l_{se} + l_{ew}$. A necessary condition for TCP pose reachability is that the shoulder point S lies no further than $l_{se} + l_{ew}$ to the TCP-offsetted-wrist-point W.

Having an equivalent spherical joint in S and a constant forearm length means that the elbow point E has to lie onto a sphere centered in S and of radius l_{se}. Similarly, E has also to lie onto a sphere centered in W and of radius l_{ew}. Therefore, E has to lie on the intersection of these two spheres. If the TCP target is too far from the robot base, *i.e.* if SW $> l_{se} + l_{ew}$, the intersection is empty, and the target is not reachable. If the target is too close, *i.e.* if SW $< l_{se} - l_{ew}$, the intersection is

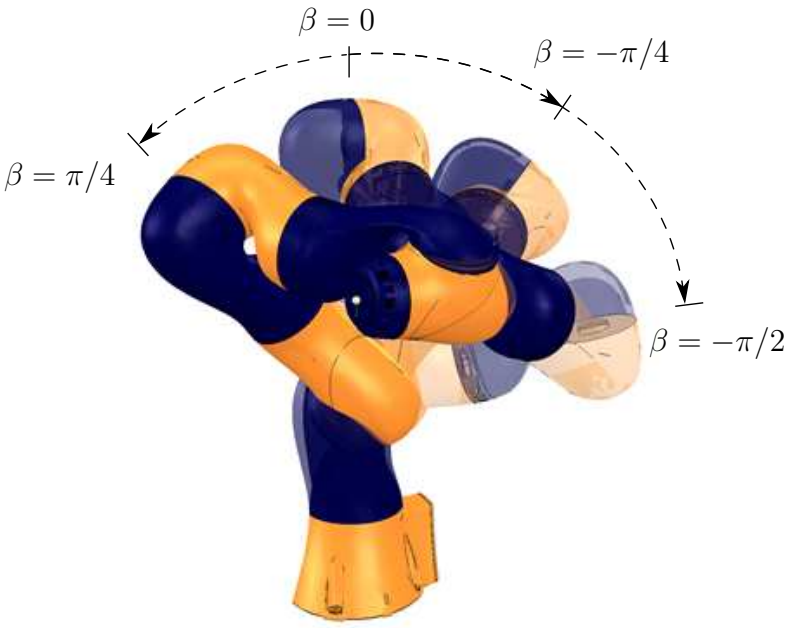


Figure 1.3.6: Elbow angle parameterisation.

also empty (the dark blue sphere of Fig. 1.3.5 entirely lies inside of the light blue one). If SW $\in [l_{se} - l_{ew}, l_{se} + l_{ew}]$, the intersection of the spheres is either a point when the pose becomes reachable or a circle in the regular case (see Fig. 1.3.4 for the regular case). This means that in the regular case, the 6-DOFs task can be performed, and the elbow can be anywhere on this circle. This freedom of the elbow about the circle can be conveniently parameterised by the *elbow angle* β (see Fig. 1.3.5), which is sometimes termed swivel angle or simply redundancy angle [39–41]. Formally, the elbow angle is defined as the angle about the shoulder-to-wrist axis between the plane defined by S, E and W and the reference plane. The reference plane is defined as the plane constructed from S, E° and W (as defined in [40]). *Note*: the right-hand-side superscript notation (.)° relates to the quantity "(.)" with the third joint angle set at zero ($q_3 = 0$).

The computation of the joint angles of the LBR iiwa with regards to a fully constraining task (6 DOFs) imposed on the tip (T) of a manipulator and an elbow angle

β is fully derived in Appendix 2. This demonstration is largely inspired from the work of Shimizu in [40], and is applied to the LBR iiwa.

1.3.3 Other sources of redundancy

Elbow Redundancy of the LBR iiwa : We described in Section 1.3.2.2.1 how the elbow angle was a way of parameterising the redundancy of the LBR iiwa arm for a positioning and orientating (6 DOFs) task of the end effector. The range of the elbow angle is exactly one lap of the elbow around the shoulder-wrist axis. Therefore it is equal to 2π, and we chose to bound it between $-\pi$ and π. The way to compute the inverse geometry with regards the elbow angle is derived in Appendix 2.

Orientation about the $\mathbf{z_{tcp}}$-axis : One other redundancy space axis that is used within this thesis is the freedom to rotate about an axis of the TCP frame (see Fig. 1.3.7). This freedom is for example justified for drilling tasks, about the drill axis. A pure rotation operator can be used in that case to change the definition of a fully constraining task for the TCP. Let us define ${}^{0}\boldsymbol{T}_{tcp}^{[a_{tcp}=a_0]}$, the transformation matrix representing a pose, arbitrarily oriented around the **z**-axis of the TCP, of the frame associated to the end effector $\mathcal{F}_{tcp}$ in the robot base reference frame $\mathcal{F}_0$. Let us define $\boldsymbol{T}_{\boldsymbol{a_{tcp}}}$, a transformation matrix only consisting of a pure rotation about the **z**-axis.

$$\boldsymbol{T}_{\boldsymbol{a_{tcp}}} = \begin{bmatrix} & \boldsymbol{r}_{\boldsymbol{a_{tcp}}} & & \mathbf{0} \\ 0 & 0 & 0 & 1 \end{bmatrix}, \text{ with}$$

$$\boldsymbol{r}_{\boldsymbol{a_{tcp}}} = \begin{bmatrix} \cos(a_{tcp}) & -\sin(a_{tcp}) & 0 \\ \sin(a_{tcp}) & \cos(a_{tcp}) & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

The redundancy parameter a_{tcp} combines with the task to form a fully constrained task ${}^{0}\boldsymbol{T}_{tcp}^{d}$ for the end effector:

$${}^{0}\boldsymbol{T}_{tcp}^{d} = {}^{0}\boldsymbol{T}_{tcp}^{[a_{tcp}=a_0]}\boldsymbol{T}_{\boldsymbol{a_{tcp}}}. \tag{1.48}$$

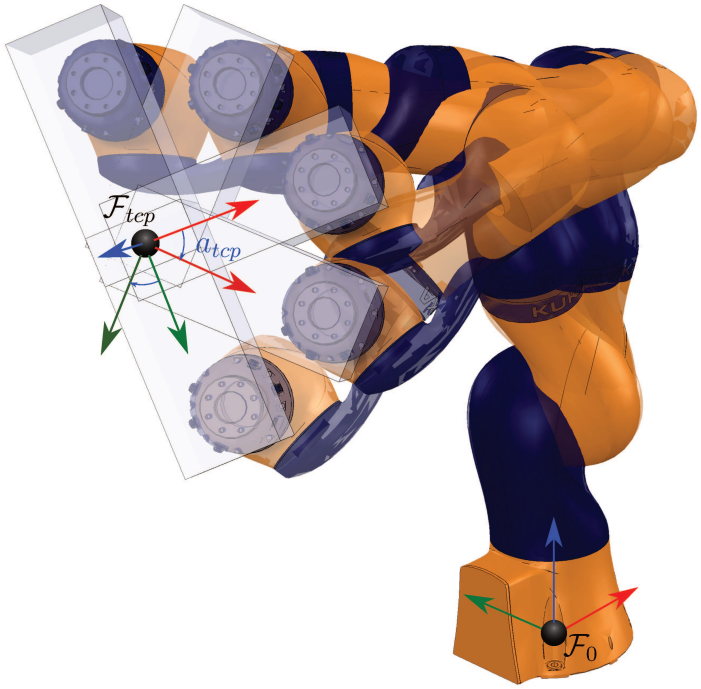


Figure 1.3.7: Redundancy parameterisation of the orientation of the TCP.

Positioning the LBR iiwa base with a KMR : Another redundancy that can easily be identified on the system consisting of a robotic arm LBR iiwa mounted on the mobile platform KMR, is the positioning and orientating of the arm robot base. Let us define x_b, y_b and θ_b the three redundancy space parameters that come from this base placement. Considering the full system, the fully constrained task is now defined with regards to a frame that is not attached to any moving part of the robot. Let us call this frame world reference frame $\mathcal{F}_w$. Now, the transformation that is produced by the arm alone can still be noted ${}^0\boldsymbol{T}_{tcp}$, but the transformation produced by the entire system is noted ${}^w\boldsymbol{T}_{tcp}$. It consists of the composition of the transformation produced by the mobile platform, which can be parameterised by the redundancy space parameters defined above, with the transformation produced by the arm alone.

$$
\begin{aligned}
{}^w\boldsymbol{T}_{tcp} &= {}^w\boldsymbol{T}_0 \, {}^0\boldsymbol{T}_{tcp} \\
&= \begin{bmatrix} \cos(\theta_b) & -\sin(\theta_b) & 0 & x_b \\ \sin(\theta_b) & \cos(\theta_b) & 0 & y_b \\ 0 & 0 & 1 & z_{ptf} \\ 0 & 0 & 0 & 1 \end{bmatrix} {}^0\boldsymbol{T}_{tcp}
\end{aligned}
$$

z_{ptf} defines the heigth of the mobile platform.

The boundaries of the redundancy space defined by the platform position in the floor plane are dependent on the target reachability. Simple geometric considerations can be used to narrow down these limits. As we saw in Section 1.3.2.2.1, for the LBR iiwa, "a necessary condition for TCP pose reachability is that the shoulder point S lies no further than $l_{se} + l_{ew}$ to the TCP-offsetted-wrist-point W". From the TCP pose, one can find the position the wrist is required to take. The shoulder point S has to lie no further than $r_{max} = l_{se} + l_{ew}$ to W. Besides, the distance between W and S cannot be smaller than when the arm is completely folded, *i.e.* when joint 4 has reached its limit $q_{4,lim}$. this distance is $r_{min} = \sqrt{{l_{se}}^2 + {l_{ew}}^2 - 2 l_{se} l_{ew} \cos(q_{4,lim})}$. Therefore, from the TCP pose, one can bound the position of the shoulder point S within the volume defined by the ball $\mathcal{B}_{max}$ bounded by sphere $\mathcal{S}_{max}$ and out of the

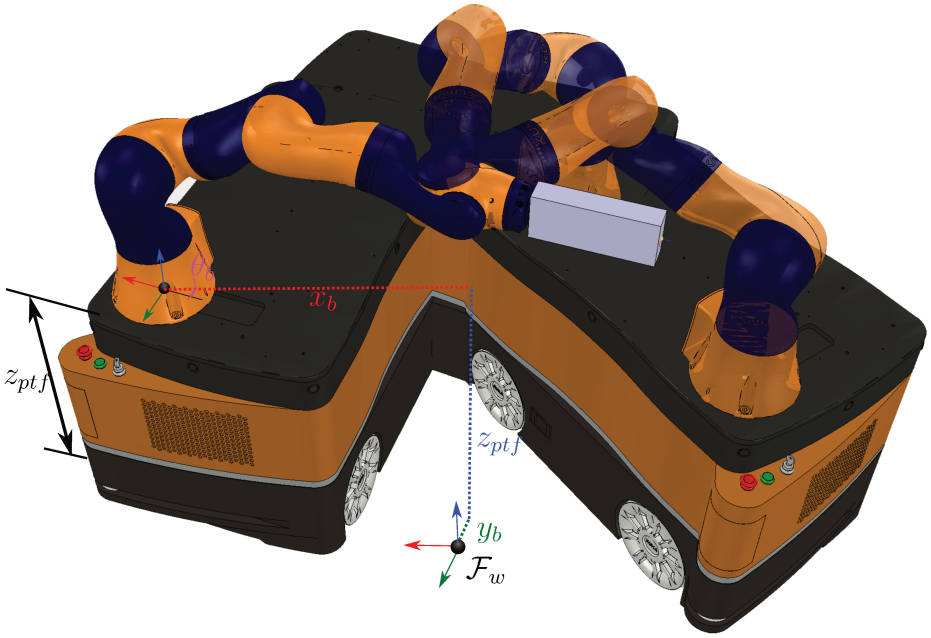


Figure 1.3.8: Positioning redundancy induced by mobile platform.

ball $\mathcal{B}_{min}$ bounded by sphere $\mathcal{S}_{min}$ (see Fig. 1.3.9). Therefore:

$$\begin{aligned} & S_{possible} \in \{P \in \mathbb{R}^3 | P \in \mathcal{B}_{max} \text{ and } P \notin \mathcal{B}_{min}\} \\ \Leftrightarrow\ & S_{possible} \in \{P \in \mathbb{R}^3 | P \in \mathcal{B}_{max} \setminus \mathcal{B}_{min}\} \end{aligned} \tag{1.49}$$

The shoulder position can also be bounded by the position of the arm base. If the LBR iiwa is mounted on the platform, we know its base frame will be z_{ptf} higher than the floor and that its **z**-axis will point upward. Therefore, the shoulder point will lie on a surface that lies $z_{ptf} + l_{bs}$ parallel to and above the floor plane, which is shown on Fig. 1.3.9 as $\mathcal{P}_{shoulder}$.

$$S_{possible} \in \{P \in \mathbb{R}^3 | P \in \mathcal{P}_{shoulder}\} \tag{1.50}$$

Therefore, combining Eq. (1.49) and Eq. (1.50) gives :

$$S_{possible} \in \{P \in \mathbb{R}^3 | P \in \mathcal{P}_{shoulder} \cap \mathcal{B}_{max} \setminus \mathcal{B}_{min}\}, \tag{1.51}$$

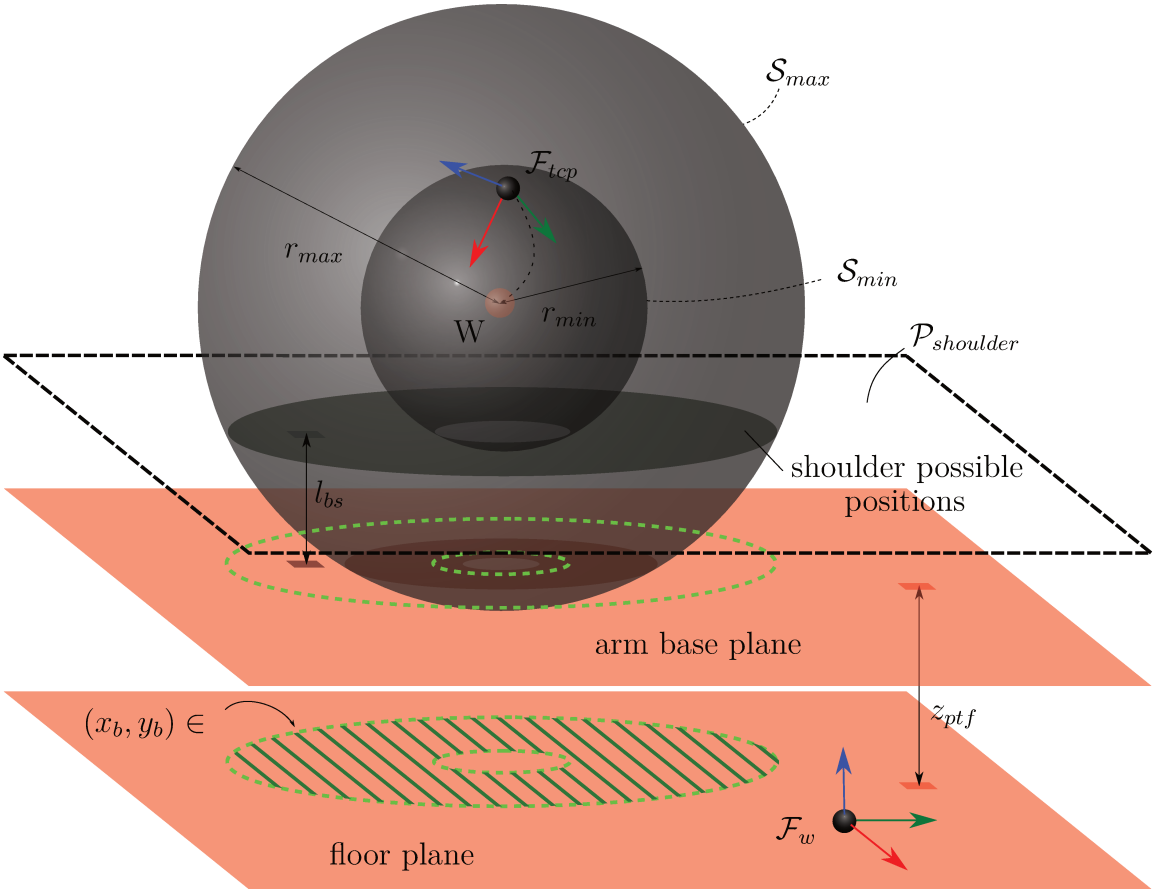


Figure 1.3.9: x_b and y_b boundaries of KMR placement.

which consequently bounds the values of x_b and y_b to the green hatched area shown on Fig. 1.3.9.

1.4 Intermediary conclusion

A framework was presented that establishes the extent of the solution space that is brought by redundancy for positional tasks.

The chapter does not provide with a systematic methodology on how to formalise redundancy spaces for any system and task. There may not always exist an easy parameterisation of redundancy at the position level, especially for hyper redundant robots, snake robots, etc... For well known and studied industrial systems, however, redundancy spaces can be formalised from which task constraints are released and what additional motion freedom extra axes provide.

The positional redundancy framework provides with multiple advantages. The boundaries of the solution space are well defined within this framework, which allows for a clear identification of the articular possibilities redundancy offers. This framework displays an intuitive and simple formulation of redundancy that is well suited to the redundancy resolution problem. Closed-form expression of the articular configurations of SRS robots, which comply with some exemplary positional tasks, were derived. Analytic solutions have the advantages, compared to numerical ones, to offer exact values, instead of approximations, and to require a lighter computational load, which are very desirable characteristics.

As a final proof of the benefits of using redundancy spaces for positional redundancy resolution, let us contextualise the use of a redundancy space framework for the optimisation of a posture dependant criteria.

Let a n-DOFs redundant robot be used for a m-DOFs positional task $\mathbf{t}$ ($n > m$). The articular configuration of the robot is denoted $\mathbf{q} = [q_1, \dots, q_n]^\intercal$. Let the function mapping an articular configuration into the corresponding value of the task[7] be called

[7] This function is often referred to as the direct geometric model of the robot.

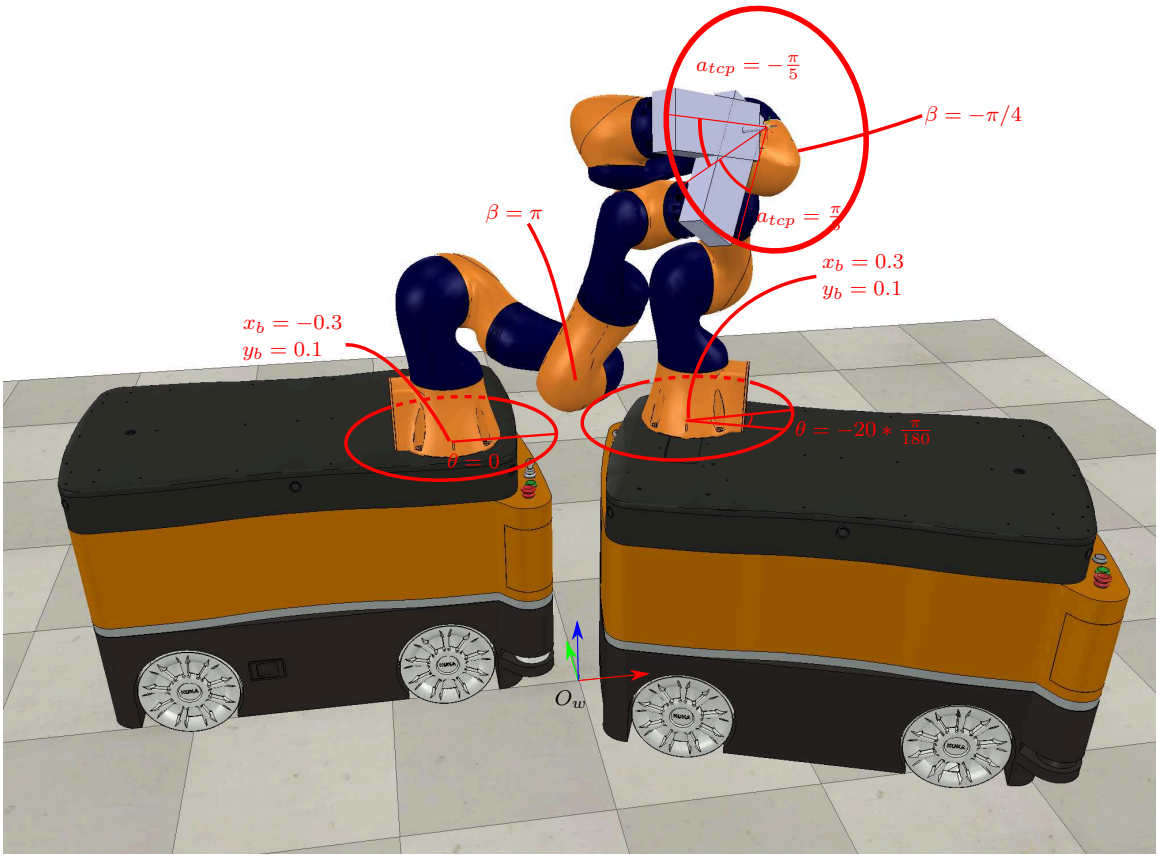


Figure 1.4.1: Two different redundancy space positions for the same end effector positional task. The end effector pose is defined with position $(0.263m, -0.422m, 0.800m)$ and with orientation in Euler XYZ convention $(1.282 \text{ rad}, 0.658 \text{ rad}, -0.991 \text{ rad} + a_{tcp}$. The redundancy space parameters are here defined as β (rad), a_{tcp} (rad), x_b (m), y_b (m) and θ_b (rad).

d, so that :

$$\begin{aligned} d :& \mathbb{R}^n \to \mathbb{R}^m \\ & \mathbf{q} \to \mathbf{t} \end{aligned} \tag{1.52}$$

Let us assume, as is the case for our use-case system and exemplary tasks, that a redundancy space formulation exists for this robot and positional task, that allows to analytically find the robot configuration $\mathbf{q}$ corresponding to the task $\mathbf{t}$ and position $\boldsymbol{\alpha} = [\alpha_1, \ldots, \alpha_n]^\intercal \in \mathbb{E} \subset \mathbb{R}^{n-m}$ in the redundancy space. Mathematically speaking, it means that there exists an analytic function g so that:

$$\begin{aligned} g :& \mathbb{R}^m \times \mathbb{E} \to \mathbb{R}^n \\ & (\mathbf{t}, \boldsymbol{\alpha}) \to \mathbf{q} \end{aligned} \tag{1.53}$$

Let C be the cost function representing the posture dependent criteria, that we want to minimise, so that :

$$\begin{aligned} C :& \mathbb{R}^n \to \mathbb{R}^+ \\ & \mathbf{q} \to C(\mathbf{q}) \end{aligned} \tag{1.54}$$

Let $C_\mathbf{t}$ be the function representing the redundancy position dependant criteria, for a given task $\mathbf{t}$. This function is an equivalent version of C, and can be defined for a given task $\mathbf{t} \in \mathbb{R}^m$ as:

$$\begin{aligned} C_\mathbf{t} :& \mathbb{E} \to \mathbb{R}^+ \\ & \boldsymbol{\alpha} \to C(g(\mathbf{t}, \boldsymbol{\alpha})) \end{aligned} \tag{1.55}$$

Now, the minimisation problem of the performance criteria can be formulated either using $\mathbf{q}$ as decision variable (scheme #1) or using $\boldsymbol{\alpha}$ as decision variable (scheme #2).

scheme #1 :

For a task $\mathbf{t}$, find $\mathbf{q}$, so that,

$$\begin{aligned} \mathbf{q} = \underset{\mathbf{q}\in\mathbb{R}^n}{\text{argmin}} \quad & C(\mathbf{q}) \\ \text{s.t.} \quad & d(\mathbf{q}) = \mathbf{t} \end{aligned}$$

scheme #2 :
For a task $\mathbf{t}$, find $\mathbf{q} = g(\mathbf{t}, \boldsymbol{\alpha})$, so that,

$$\boldsymbol{\alpha} = \underset{\boldsymbol{\alpha}\in\mathbb{E}}{\text{argmin}} \quad C_{\mathbf{t}}(\boldsymbol{\alpha})$$

Scheme #2 presents some major advantages over scheme #1:

- In scheme #1, the constraint equation $d(\mathbf{q}) = \mathbf{t}$ requires to find a configuration that complies with the tasks before testing the value of $C(\mathbf{q})$. On the other hand, the optimisation problem is unconstrained in scheme #2. In scheme #2, changing the value of α will not hamper the fulfilment of positional task $\mathbf{t}$. Scheme #2 can be seen a direct search in the constrained space of configurations complying with the task, which is naturally parameterised by the redundancy space parameters.
- The search space of scheme #2 has a smaller dimension that the search space of scheme #1, which suggests a much lighter computational load to find a solution.

Given these advantages, the framework of redundancy space will be used in chapters Chapter 2 and Chapter 3 for redundancy resolution purposes. In these chapters, the posture dependent criteria which will be introduced are related to the stiffness of serial systems, and their ability to counter a spatial force.

2

Deformational behaviour improvement of serial redundant manipulators

The theoretical knowledge of the misplacement of the end effector caused by an interaction force is a key notion where accurate and high quality robotised operations are concerned. In this chapter, the Cartesian compliance matrix of serial robots is first introduced. This tool is analysed for the LBR iiwa robot, and is then used to predict the misplacement of the end effector when under the influence of a force, which typically happens during machining operations. Redundant manipulators have the ability to perform a position level task in a continuous space of articular positions. It so happens that the Cartesian rigidity of serial manipulators is strongly related to the posture used to perform a task. Therefore, exploiting positional redundancy is an interesting way of modifying the deformational behaviour of the robot without affecting the end-effector placement.

In this chapter, the framework of redundancy spaces (Section 1.3) is used to enhance the performance related to stiffness and accuracy of redundant serial robots sustaining an interaction force.

Contents

2.1 Industrial context - Robotised machining industry driven by accuracy

Figure 2.1.1: A typical assembly interface between two fuselage panels involving hundreds of holes.

Ideally, repetitive high value added tasks such as milling or drilling are given to CNC machines. While these systems are very accurate and efficient, they are very expensive, their workspace volumes are generally small, and they can't be easily moved around. Large parts such as aircraft panels or wings cannot be fitted into them, and many drilling operations require to be directly followed by assembly operations to maintain multi layered hole alignment and perfect parts matching. In automotive or aircraft production factories, a lot of drilling operations are still performed by human workers, who can't always guaranty high repeatability and quality without using very onerous and cumbersome special toolings. Therefore, drilling and milling serial robotic systems have recently appeared on this market. While most robotised operations are pick and place, welding or assembly tasks (approximately 75%), man-

ufacturers are now willing to extend robot work to higher value added, and quite irreversible operations such as drilling or milling. More than one million holes can be counted on good sized commercial planes. This constitutes a windfall of cost cut opportunities for aircraft manufacturers, who intensely look into high quality-low cost robotised solutions. However, a drawback of these versatile serial systems is their low accuracy and rigidity. Paper [43] reports that standard serial systems have a 1N/μm stiffness against 50 N/μm for less versatile standard CNC machines. This is a major issue in operations requiring strong physical interactions with the environment, as it may lead to poor machining surface quality or unacceptable inaccuracies due to mechanical deformation. Advances in actuation and sensing have recently brought small collaborative robots on center stage. While these systems are even more compact and versatile, their rigidity is even lower than the one of traditional serial robots (our estimation shows it is often more than twenty times lower for the LBR iiwa than on traditional serial systems.), which further complicates the problem of having high quality drilling or milling operations.

This chapter presents a practical use of the redundancy space of serial manipulators to improve the matter of low stiffness causing positioning discrepancies. It characterises the rigidity of the LBR iiwa and shows how redundancy can be exploited to enhance its rigidity characteristics and accuracy under the strain of a drilling task.

2.2 Stiffness analysis in the literature

For most industrial robots, joints compliance and more specifically gears compliance is known to be a source of deformation and discrepancy during robot motion or robot-environment interaction [44–46]. A number of methods have been developed in the literature to improve machining accuracy. Some propose, for milling applications, to modify the initial path in order to account for the low-stiffness-linked discrepancies. The authors of [47] choose to correct the path by measuring the machining surface after an initial trial at milling. In [48], the authors first identify the stiffness of a milling robot. Then, they pre-compute a path deviation, based on the estimated

Figure 2.1.2: Cartesian rigidity orders of magnitude for different systems [42].

geometric discrepancies that would happen along the path, by applying a theoretical milling wrench of the end effector.

Other methods suggest to change the posture of the robot, to improve machining accuracy. Underlying these methods is the notion of kinematic redundancy, which allows multiple postures for the same problem. In [49], the robot base location is chosen to maximise the volume of the kinematic and dynamic ellipsoids ([50]) of the system along a milling path. Additionally, the initial position of the end effector is chosen to minimise the torque variations along the path. Given the complexity of the problem, the others use a genetic algorithm strategy. Guo (in [51]) and Bu (in [42]) suggest to use, for drilling applications involving 6 DOFs robots, the redundancy that exists on the end effector orientation about the drill axis to optimise criteria. Guo introduces a performance index which is based on the volume of the compliance ellipsoid, which stands for an overall compliance score. We may however argue that, by mixing translation and rotation information, the performance index introduced therein have no clear physical meaning, as was notified in [52]. Bu, instead chooses to ignore the rotational components of the compliance matrix, claiming that orientational errors are negligible for his system (below 0.05 mm for a 1000 N force applied at the end effector). The strategy described therein focuses on the quality of the countersink, which relates directly to the compliance along the drilling direction. It is this specific compliance that is sought to be minimised. Additionally, Bu presents an interesting analysis of the effect of the deformation of the drill and pre-load of the end effector (sometimes called clamp) onto the perceived rigidity of the interaction.

The work presented in this chapter also takes roots in the analysis of the compliance matrix of the robot. One of the differences here is that, while the notion of a freedom on the posture was suggested in many of the works presented above, the exploitation of kinematic redundancy was never explicitly mentioned. The main idea remains the same, but the concept of redundancy brings a little more genericity and allows more transposability to other systems. The framework of redundancy space introduced in Section 1.3 will be used to demonstrate the applicability of the method to a system that is kinematically redundant of order two for the task of drilling (the LBR iiwa).

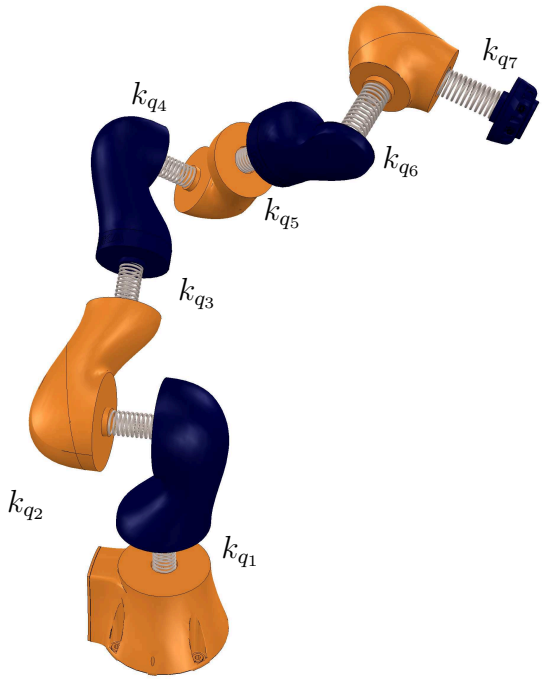


Figure 2.2.1: Compliance model concentrated in joints as torsion springs.

Additionally, while we know compliance and stiffness to be very interesting concepts, we choose to present the results in terms of deformation of the task. This choice is motivated by the fact that mechanical tolerances are given in metres and not Newton per meters, which yields more palpable and intuitive results. Finally, the entire method is applied to a system that has very rarely been used for drilling tasks. The stiffnesses recorded for this system are low, and this makes it an even more challenging problem.

As suggested in [42, 46, 51, 53], the joints radial deformations, as well as the links deformations are neglected and a joint compliance is modelled as a torsion spring with stiffness value $k_{q,i}$ for each joint i. We will assume for the time being that the end effector and all the links are undeformable rigid bodies. Additionally, we will assume the drilling process does not involve any feeding motion from the manipulator's part. The feeding and turning motions of the drill are performed by the end effector, and the drilling process is assumed to be under stable condition.

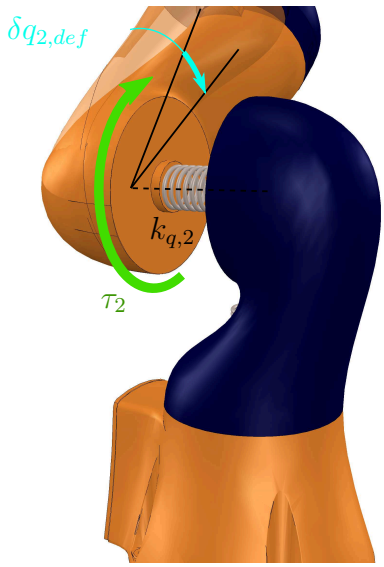


Figure 2.3.1: Example of joint model (joint 2). The ith joint compliance is modelled as a torsion spring with stiffness $k_{q,i}$.

2.3 Cartesian compliance model of a serial system

Assuming an external spatial force $-\mathbf{f}$ is applied to the end effector of a N-DOFs serial robot. To maintain a static posture, the end effector must itself apply a spatial force $\mathbf{f}$ onto its environment. Given the kinetostatic Eq. (1.19), and the robot geometric Jacobian matrix (see Section 1.1.4.3) $\boldsymbol{J}$, the torque vector $\boldsymbol{\tau}$, needed to apply this force is equal to:

$$\boldsymbol{\tau} = \boldsymbol{J}^{\intercal}\mathbf{f}.$$

Now, looking at the deformation law of the joints illustrated in Fig. 2.3.1, the articular deformation vector $\boldsymbol{\delta}\mathbf{q} = \begin{bmatrix}\delta q_{1,def} & \dots & \delta q_{N,def}\end{bmatrix}^{\intercal}$ can be expressed as :

$$\boldsymbol{\delta}\mathbf{q} = {\boldsymbol{K}_q}^{-1}\boldsymbol{\tau}, \tag{2.1}$$

Where

$$\boldsymbol{K}_q = \begin{bmatrix} k_{q,1} & & (0) \\ & \ddots & \\ (0) & & k_{q,N} \end{bmatrix} \tag{2.2}$$

is the diagonal positive articular rigidity matrix of the N-DOFs robot with joint stiffnesses $k_{q,i}$ on the main diagonal. The static joints stiffness values of a robot can be experimentally determined by measuring each joint angular variation to a set of imposed external forces or torques. The Cartesian displacement $\boldsymbol{\delta}\mathbf{X}$ of the end effector caused by a local joint displacement $\boldsymbol{\delta}\mathbf{q}$ is given by

$$\boldsymbol{\delta}\mathbf{X} = \boldsymbol{J}\boldsymbol{\delta}\mathbf{q}. \tag{2.3}$$

Therefore, the Cartesian displacement (twist) caused by the application of an external spatial force at the end effector can be expressed by combining these equations. This gives us the expression of the Cartesian compliance matrix of the robot $\boldsymbol{C}$

$$\boldsymbol{\delta}\mathbf{X} = \boldsymbol{C}\mathbf{f} = (\boldsymbol{J}\boldsymbol{K}_q{}^{-1}\boldsymbol{J}^{\intercal})\mathbf{f}. \tag{2.4}$$

To better understand the composition of this matrix, it can be interesting to isolate translational from orientational terms. The Cartesian displacement can be divided into a translational displacement $\boldsymbol{\delta}\mathbf{p}$ and a rotational displacement $\boldsymbol{\omega}$. Similarly, the spatial force can be decomposed into a translational component $\mathbf{F}$ and a moment $\mathbf{M}$:

$$\boldsymbol{\delta}\mathbf{X} = \begin{bmatrix} \boldsymbol{\delta}\mathbf{p} \\ \boldsymbol{\omega} \end{bmatrix} \text{ and}$$
$$\mathbf{f} = \begin{bmatrix} \mathbf{F} \\ \mathbf{M} \end{bmatrix}.$$

With these notation, the compliance matrix can be divided into four 3×3 submatrices

$\boldsymbol{C}_{\mathrm{PF}}$, $\boldsymbol{C}_{\mathrm{PM}}$, $\boldsymbol{C}_{\omega\mathrm{F}}$ and $\boldsymbol{C}_{\omega\mathrm{M}}$ as

$$\boldsymbol{C} = \begin{bmatrix} \boldsymbol{C}_{\mathrm{PF}} & \boldsymbol{C}_{\mathrm{PM}} \\ \boldsymbol{C}_{\omega\mathrm{F}} & \boldsymbol{C}_{\omega\mathrm{M}} \end{bmatrix}.$$

In this formulation, $\boldsymbol{C}_{\mathrm{PF}}$ (terms in m/N) refers to the translational compliance of the system with regards to translational forces, $\boldsymbol{C}_{\mathrm{PM}}$ (terms in m/Nm) refers to the translational compliance of the system with regards to moments, $\boldsymbol{C}_{\omega\mathrm{F}}$ (terms in rad/N) refers to the orientational compliance of the system with regards to translational forces, and $\boldsymbol{C}_{\omega\mathrm{M}}$ (terms in rad/Nm) refers to the orientational compliance of the system with regards to moments. Quite remarkably, one can note that $\boldsymbol{C}_{\mathrm{PM}} = \boldsymbol{C}_{\omega\mathrm{F}}{}^{\intercal}$. This can be clarified by expanding each of these matrices in terms of the translational and rotational parts of the Jacobian matrix $\boldsymbol{J} = \begin{bmatrix} \boldsymbol{J}_{\mathrm{X}}{}^{\intercal} & \boldsymbol{J}_{\omega}{}^{\intercal} \end{bmatrix}^{\intercal}$. Doing this, one will find that $\boldsymbol{C}_{\mathrm{PM}} = \boldsymbol{J}_{\mathrm{X}}\boldsymbol{K}_q{}^{-1}\boldsymbol{J}_{\omega}{}^{\intercal}$ and that $\boldsymbol{C}_{\omega\mathrm{F}} = \boldsymbol{J}_{\omega}\boldsymbol{K}_q{}^{-1}\boldsymbol{J}_{\mathrm{X}}{}^{\intercal} = (\boldsymbol{J}_{\mathrm{X}}\boldsymbol{K}_q{}^{-\intercal}\boldsymbol{J}_{\omega}{}^{\intercal})^{\intercal} = (\boldsymbol{J}_{\mathrm{X}}\boldsymbol{K}_q{}^{-1}\boldsymbol{J}_{\omega}{}^{\intercal})^{\intercal}$.

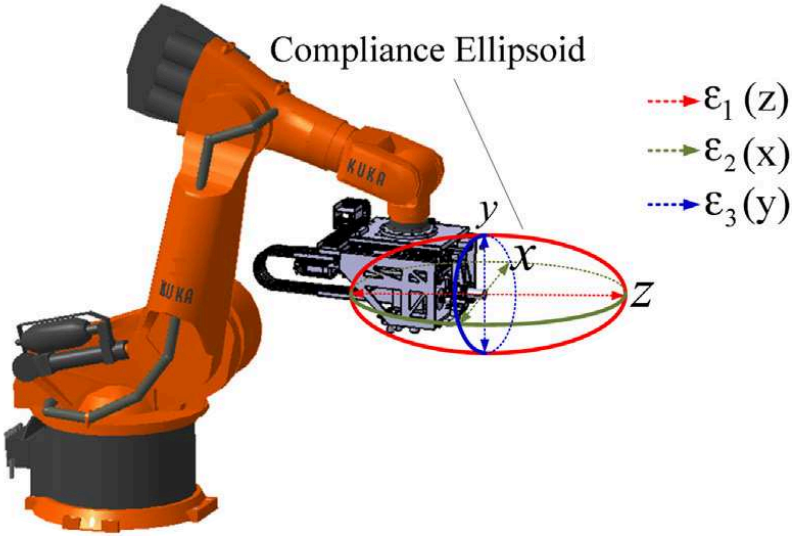


Figure 2.3.2: $\boldsymbol{C}_{\mathrm{PF}}$: Translational displacement under translational force compliance ellipsoid (figure credits go to Bu et al. in [42]).

Bu, in [42], reports that the rotational displacements influencing the drilling normal accuracy are negligible for his KR500 robot. His computations show that the drill axis angular deviation is inferior to 0.03°, when under a 1000 N pressing force at the end effector, which is far below the usual 0.5° mechanical tolerance that is commonly seen in aircraft or automotive production. Bu thus simplifies the study of the compliance matrix by only focusing on submatrix $\boldsymbol{C}_{\mathrm{PF}}$. He then builds the Cartesian compliance ellipsoid by computing the set of translational forces leading to an overall displacement norm $\sqrt{\boldsymbol{\delta}\mathbf{p}^{\intercal}\boldsymbol{\delta}\mathbf{p}}$ equal to one.

$$\begin{aligned} &\boldsymbol{\delta}\mathbf{p}^{\intercal}\boldsymbol{\delta}\mathbf{p} = 1 \\ &\Rightarrow \mathbf{F}^{\intercal}\boldsymbol{C}_{\mathrm{PF}}{}^{\intercal}\boldsymbol{C}_{\mathrm{PF}}\mathbf{F} = 1 \end{aligned}$$

By computing the eigenvalues and eigenvectors of matrix $\boldsymbol{C}_{\mathrm{PF}}{}^{\intercal}\boldsymbol{C}_{\mathrm{PF}}$, one can then visualise in 3-dimensional Euclidean space the directions in which translational forces have the most (semi-minor axis) or the least (semi-major axis) influence over the displacement of the end effector.

2.4 Compliance matrix and related accuracy measures applied to the LBR iiwa

In the case of the LBR iiwa, however, the orientational deviation of the drill axis is often not negligible compared to the usual mechanical tolerances. The following paragraphs will describe a procedure which assesses the values of the misplacement of the end-effector of this robot, thus providing with an order of magnitude.

The embedded joint torque sensors of the 7-DOFs Kuka LBR iiwa can be advantageously exploited to simplify the joint stiffness identification procedure, as detailed in [54]. Given the assumption saying that only deformations happening within the joints are taken into account, the reflected stiffness values for one joint is the result of the serial combination of the gears stiffness value k_{gear} and the equivalent control stiffness value k_{ctrl}. These serial stiffnesses add up to form an equivalent overall stiff-

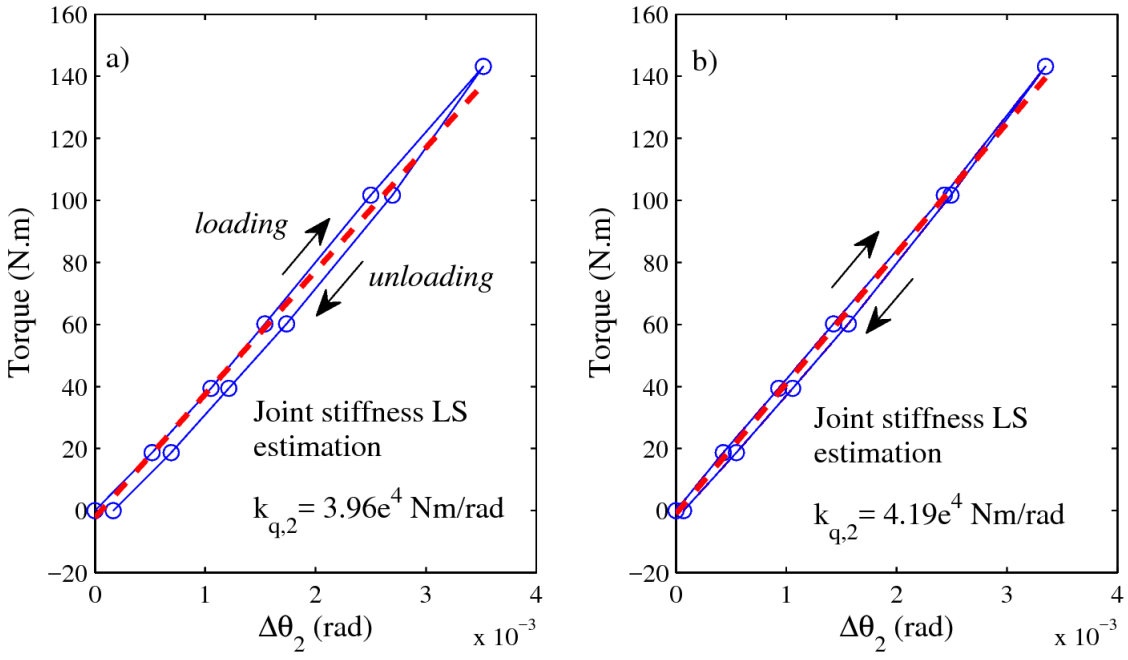


Figure 2.4.1: Deflection of joint 2 under static increasing and decreasing payloads. a) under position control mode, b) when the motor brakes are locked (gear stiffness only)

ness $k_{eq} = \frac{k_{gear}k_{ctrl}}{k_{gear}+k_{ctrl}}$. Therefore, the joint stiffness values are dependent on the robot control mode. Fig. 2.4.1 shows the exploitation of the experimental measurements leading to the evaluation of joint 2 stiffness in position control mode (a) and in locked-joint-brake mode (b) for the Kuka LBR iiwa 14 R820. The first setup evaluates the joint stiffness with the two serial stiffnesses ($k_{eq} = \frac{k_{gear}k_{ctrl}}{k_{gear}+k_{ctrl}}$) while the second only measures the gear stiffness $k_{eq} = k_{gear}$. The results seem to show that the two setups yield very similar overall joint stiffnesses. This leads to the conclusion that the gear stiffness is much smaller than the one coming from the control law. Therefore, if $k_{gear} \ll k_{ctrl}, k_{eq} = \frac{k_{gear}k_{ctrl}}{k_{gear}+k_{ctrl}} \simeq \frac{k_{gear}k_{ctrl}}{k_{ctrl}} = k_{gear}$. The measured values of the 7 joint stiffness in position control mode, taken from [54], are given in Table 2.4.1. They are used as a basis for the work introduced in this chapter, since they are very similar to the actual values of the gear stiffnesses.

$k_{q,1}$	$k_{q,2}$	$k_{q,3}$	$k_{q,4}$	$k_{q,5}$	$k_{q,6}$	$k_{q,7}$
4.2e4	4.0e4	1.2e4	2.0e4	0.80e4	0.38e4	0.48e4

Table 2.4.1: Identified joint stiffness values of the Kuka LBR iiwa 14R820 (Nm/rad) taken from [54]

Use case setup description : Let $\mathbf{f}$ be a force corresponding to the drilling process.Let T be a point located at the positional offset (0.0 m, −0.175 m, 0.0 m). Let the TCP reference frame $\mathcal{F}_{tcp}$ be positioned in T, with an orientational offset (in terms of Euler angle ZYX convention) (0.0 rad, 0.0 rad, $-\pi/2$ rad) from $\mathcal{F}_7$. This TCP and its Cartesian axes with be labelled "*unloaded*" to refer to the state of this frame when the system doesn't counter any external force and "*loaded*" when a force is applied to the end-effector. Denoting ${}^{tcp}\mathbf{f}$ the numerical expression of $\mathbf{f}$ in $\mathcal{F}_{tcp}$, let us assume that

$$ {}^{tcp}\mathbf{f} = \begin{bmatrix} 0N & 0N & -100N & 0Nm & 0Nm & -5Nm \end{bmatrix}^{\mathsf{T}}. $$

It corresponds to a force of 100 N along and a torque of −5 Nm about the $\mathbf{z}_{tcp}$-axis (the drill axis). Let us denote ${}^{tcp}\boldsymbol{J}$ the geometric Jacobian matrix at point T, expressed in $\mathcal{F}_{tcp}$. The Cartesian compliance matrix of the system at point T, expressed in $\mathcal{F}_{tcp}$, can be derived thanks to Eq. (2.4).

$$ {}^{tcp}\boldsymbol{C} = {}^{tcp}\boldsymbol{J}\,{\boldsymbol{K}_q}^{-1}{}^{tcp}\boldsymbol{J}^{\mathsf{T}} \tag{2.5} $$

Let the current configuration of the LBR iiwa be : $q1 = 110°$, $q2 = 47°$, $q3 = -62°$, $q4 = 108°$, $q5 = -158°$, $q6 = -15°$ and $q7 = -41°$.

Cartesian stiffness computations : The Cartesian compliance matrix ${}^{tcp}\boldsymbol{C}$ ex-

pression, for the current configuration can be written as:

$${}^{tcp}\boldsymbol{C} = \begin{bmatrix} \boldsymbol{C}_{\mathrm{PF}} & \boldsymbol{C}_{\mathrm{PM}} \\ \boldsymbol{C}_{\omega\mathrm{F}} & \boldsymbol{C}_{\omega\mathrm{M}} \end{bmatrix}, \text{ where}$$

$$\boldsymbol{C}_{\mathrm{PF}} = \begin{bmatrix} 3.746 & 0.6305 & -0.7063 \\ 0.6305 & 2.691 & -2.458 \\ -0.7063 & -2.458 & 2.59 \end{bmatrix} \quad \cdot 10^{-5} \text{ m/N}$$

$$\boldsymbol{C}_{\mathrm{PM}} = \begin{bmatrix} -3.392 & 5.861 & -5.627 \\ -6.707 & 0.8619 & -2.618 \\ 6.708 & -1.022 & 2.53 \end{bmatrix} \quad \cdot 10^{-5} \text{ m/Nm}$$

$$\boldsymbol{C}_{\omega\mathrm{F}} = \begin{bmatrix} -3.392 & -6.707 & 6.708 \\ 5.861 & 0.8619 & -1.022 \\ -5.627 & -2.618 & 2.53 \end{bmatrix} \quad \cdot 10^{-5} \text{ rad/N}$$

$$\boldsymbol{C}_{\omega\mathrm{M}} = \begin{bmatrix} 21.63 & -6.868 & 10.72 \\ -6.868 & 29.17 & 6.85 \\ 10.72 & 6.85 & 26.9 \end{bmatrix} \quad \cdot 10^{-5} \text{ rad/Nm}$$

A drilling force ${}^{tcp}\mathbf{f}$ applied at the end-effector's TCP leads to its Cartesian displacement (expressed in $\mathcal{F}_{tcp,unloaded}$, which corresponds to the unloaded pose):

$$\boldsymbol{\delta}\mathbf{X} = \begin{bmatrix} \boldsymbol{\delta}\mathbf{p}^{\mathsf{T}} & \boldsymbol{\omega}^{\mathsf{T}} \end{bmatrix}^{\mathsf{T}} \text{ where,} \tag{2.6}$$

$$\boldsymbol{\delta}\mathbf{p} = \begin{bmatrix} -0.9876 \\ -2.588 \\ 2.716 \end{bmatrix} \quad \text{mm}$$

$$\boldsymbol{\omega} = \begin{bmatrix} 7.243 \\ -0.6798 \\ 3.875 \end{bmatrix} \quad \times 10^{-3} \text{ rad}$$

Some deformational behaviour indices : The translational displacement of the TCP is directly given by the value of $\boldsymbol{\delta}\mathbf{p}$. We can determine the imprecision in the drilling plane positioning by looking at $\delta x = \boldsymbol{\delta}\mathbf{p}(1)$ and $\delta y = \boldsymbol{\delta}\mathbf{p}(2)$. The displacement along $\mathbf{z}_{tcp,unloaded}$, $\boldsymbol{\delta}\mathbf{p}(3)$, is also very important as it goes opposite to the feed motion.

The orientation shift of the drill axis before and after the application of the force corresponds to the angle $\delta\theta$ lying between vector $\mathbf{z}_{tcp,unloaded}$ and vector $\mathbf{z}_{tcp,loaded}$. To compute just this, we can first compute the rotation matrix corresponding to the angle axis representation whose value is $\boldsymbol{\omega}$. This matrix is a representation of the displaced TCP axes within the non-moved TCP reference frame. The angle axis notations will be :

$$\begin{aligned} \theta &= \|\boldsymbol{\omega}\| \\ w_x &= \boldsymbol{\omega}(1)/\theta \\ w_y &= \boldsymbol{\omega}(2)/\theta \\ w_z &= \boldsymbol{\omega}(3)/\theta \end{aligned} \tag{2.7}$$

With abbreviations $c_\theta = \cos(\theta)$, $s_\theta = \cos(\theta)$ and $v_\theta = 1 - \cos(\theta)$, one can compute the value of the $\boldsymbol{\omega}$ angle axis equivalent rotation matrix[1]:

$$^{tcp,unloaded}\boldsymbol{R}_{tcp,loaded} = \begin{bmatrix} {w_x}^2 v_\theta + c_\theta & w_x w_y v_\theta - w_z s_\theta & w_x w_z v_\theta + w_y s_\theta \\ w_x w_y v_\theta + w_z s_\theta & {w_y}^2 v_\theta + c_\theta & w_y w_z v_\theta - w_x s_\theta \\ w_x w_z v_\theta - w_y s_\theta & w_y w_z v_\theta + w_x s_\theta & {w_z}^2 v_\theta + c_\theta \end{bmatrix} \tag{2.8}$$

Computing the angle between $\mathbf{z}_{tcp,unloaded}$ and $\mathbf{z}_{tcp,loaded}$ comes down to determining

[1] Eq. (2.8) corresponds to the expansion of the matrix that was introduced in Eq. (4.8) for the elbow angle rotation matrix

$\delta\theta$ as:

$$\delta\theta = \arccos\left(\begin{bmatrix}0\\0\\1\end{bmatrix} \cdot \begin{bmatrix}w_x w_z v_\theta + w_y s_\theta\\ w_y w_z v_\theta - w_x s_\theta\\ {w_z}^2 v_\theta + c_\theta\end{bmatrix}\right) = \arccos({w_z}^2 v_\theta + c_\theta) \tag{2.9}$$

Deformational behaviour study : For our current use case and configuration, the drill shift angle equals $\delta\theta = 0.42°$. A broader study, taking 10000 randomly generated articular configurations, shows some relevant displacements caused by ${}^{tcp}\mathbf{f}$ in Fig. 2.4.2. Looking at these results, the hypothesis used in [42], according to which the orientational shift is negligible against the mechanical tolerances seen in production factories (0.5°), doesn't hold with the LBR iiwa. Therefore, the simplifications performed for the exploitation of the compliance matrix, consisting of ignoring the orientational terms, don't hold with the robot we use.

Consequences on the use of stiffness ellipsoids for the LBR iiwa : To keep the framework of compliance ellipsoids (used by Bu [42]) with the LBR iiwa, one could produce an overall (translational and orientational) ellipsoid. However, this would involve non physical mixes of units. Indeed, the expression of the ellipsoid would involve finding spatial forces of unitary euclidean norm. This is already physically irrelevant as spatial forces mix N (Newtons) and Nm (Newton meters). Additionally, doing this would involve the computation of $\boldsymbol{C}^\intercal\boldsymbol{C}$ which in time also involves non physical mixes of physical quantities ($\boldsymbol{C}_{\mathrm{PF}}{}^\intercal\boldsymbol{C}_{\mathrm{PF}} + \boldsymbol{C}_{\omega\mathrm{F}}{}^\intercal\boldsymbol{C}_{\omega\mathrm{F}}$ adds up terms of unit $(\mathrm{m/N})^2$ with terms of unit $(\mathrm{rad/(Nm)})^2$).

Another solution would be to produce one ellipsoid for each of the identified submatrices of $\boldsymbol{C}$. The results related to orientational shifts, however, would be very difficult to interpret as they are. The combined analysis and optimisation of all these ellipsoids for redundancy resolution could also prove a very tedious.

To remain physically consistent and keep the interpretation simple, the strategy we decide to use is to directly compute the deformation of the system under the stress of a force. Additionally to being physically consistent, we feel that using displacements

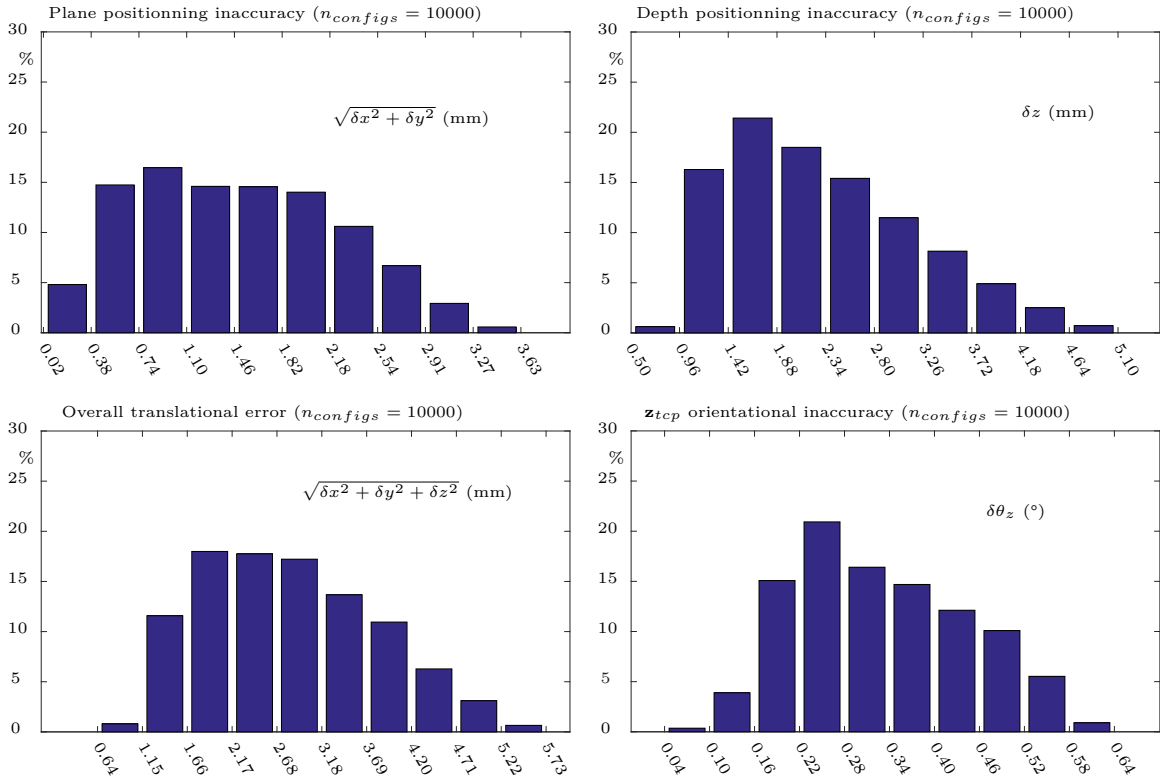


Figure 2.4.2: Study on the TCP misplacements of an LBR iiwa under a static force ${}^{tcp}\mathbf{f} = \begin{bmatrix} 0 & 0 & -100 & 0 & 0 & -5 \end{bmatrix}^\intercal$.

instead of compliance/stiffness is a more pragmatic and understandable result for end users. It is also easier to use in the context of mechanical tolerances.

2.5 Redundancy exploitation of the LBR iiwa for rigidity and accuracy enhancement

Use case context : Let us consider a drilling application. Let us assume our setup to be a drilling panel, a LBR iiwa manipulator and a drilling end effector. Let us assume that this end effector actuates two independent motions. These two motions are the feed of the drill[2] and its rotation. Assuming the TCP and input force ${}^{tcp}\mathbf{f}$ are the same as in the previous use case (see Section 2.4).

Redundancy space considerations : From a process point of view, this drilling task requires 5-DOFs. Three DOFs are required for positioning the drill two others for orientating its feed axis (the **z**-axis of the drill is free). 3-DOFs are generally required to orientate a TCP, but in this case, the rotation about the drill axis is free. Therefore, in this context, the LBR iiwa, with its 7 independent actuated joints, naturally offers two redundancies. The elbow angle β (see Fig. 1.3.6) and the drill axis orientation a_{tcp} (see Fig. 1.3.7) can be used to parameterise the redundancy space of the robot. We can describe this positional task thanks to the position of the TCP and its orientation thanks to the Euler-XYZ convention. The Euler-XYZ convention sets the orientation of the **z**-axis last, and we will therefore let it free. The positional task can therefore be defined here as a position $(-0.506 \text{ m}, -0.241 \text{ m}, 0.586 \text{ m})$ and **z**-free orientation $(1.2179 \text{ rad}, 0.8611 \text{ rad}, a_{tcp})$

Objective : Once the posture of the robot complying with the positional task will be reached, the manipulator will have to maintain it as best as possible while the drilling process will be running. We saw that the posture of the robot influences tremendously its deformational behaviour. Therefore, the objective here is to choose

[2]The feed corresponds to the translational displacement of the drill that is used to enter the drilling panel.

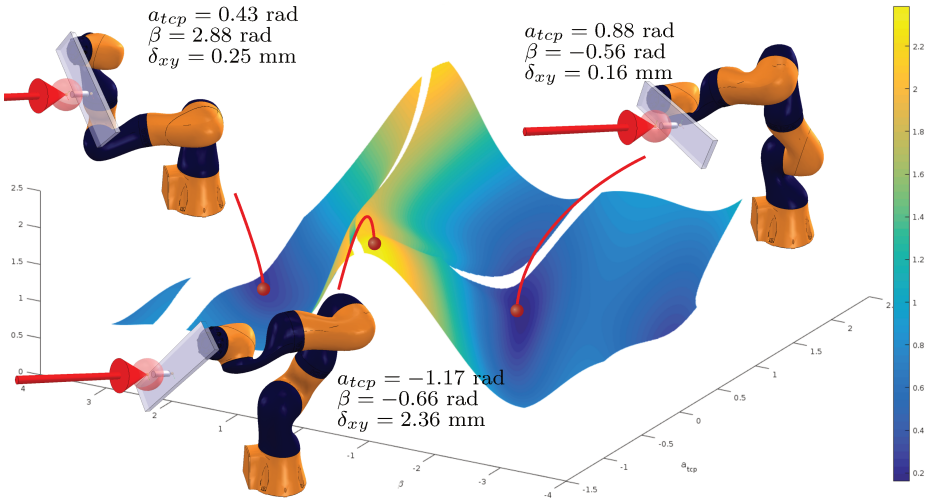


Figure 2.5.1: Drilling plane displacement caused by a spatial force ${}^{tcp}\mathbf{f} = \begin{bmatrix} 0N & 0N & -100N & 0Nm & 0Nm & -5Nm \end{bmatrix}^{\mathsf{T}}$ applied at the TCP, with regards to the two redundancy parameters β (elbow angle) and a_{tcp} (drill orientation) considered. The posture associated to three noticeable points of the redundancy spaces are displayed along with the variations of δ_{xy}.

a posture for the system that will provide sufficient accuracy characteristics when under the influence of a force. Three criteria will be used in this example[3] :

- The drill plane displacement is given by

$$\delta_{xy} = \sqrt{\delta x^2 + \delta y^2}.$$

- The axial shift of the drill (depth positioning inaccuracy) is given by δz.
- The orientational misplacement of the drill axis is given by

$$\delta\theta_z = \arccos({w_z}^2 v_\theta + c_\theta).$$

To find this posture, a search is made within the solution space of the positional task,

[3]They were introduced in Section 2.4

which is described by the redundancy space parameterised by (a_{tcp}, β).

Results of the positional redundancy resolution : Fig. 2.5.1 shows the variations of the drill plane misplacement over the redundancy space of the robot. Each point of the (a_{tcp}, β) plane corresponds to a different posture of the manipulator that complies with the positional task. The variations of the planar misplacement are within the range $[0.16 \text{ mm}, 2.36 \text{ mm}]$ for this positional task. These drastic variations clearly illustrate the posture-dependency of the accuracy of the system and the potential benefits an enlightened redundancy management may bring. An optimisation scheme may be used on the value of the planar misplacement in order to find the posture which minimises this criterion.

However, the planar misplacement may not be the only criterion we want to minimise. If we also want to avoid a substantial axial misalignment and axial shift of the drill, we may want to ensure that all three of these misplacements are contained within user-chosen tolerance boundaries. Fig. 2.5.2 displays in parallel the planar displacement, the displacement along the drill axis, and the orientational error on the drill axis. Each of these plots is overlayed with hatched areas representing regions of the redundancy space which don't comply with the tolerances of the associated criterion. Given the union of the tolerance-respecting areas, the available redundancy space that is within all the misplacements tolerances is displayed in the blue non-hatched area showed on the bottom right of the figure.

2.6 Intermediary Conclusion

This chapter presented a strategy aiming at characterising and improving the deformational behaviour of a redundant serial robot under the stress of a force. The strategy exploited the positional redundancy of the system with regards to a given task and an interaction force, to find the subset of postures reducing the misplacement of the

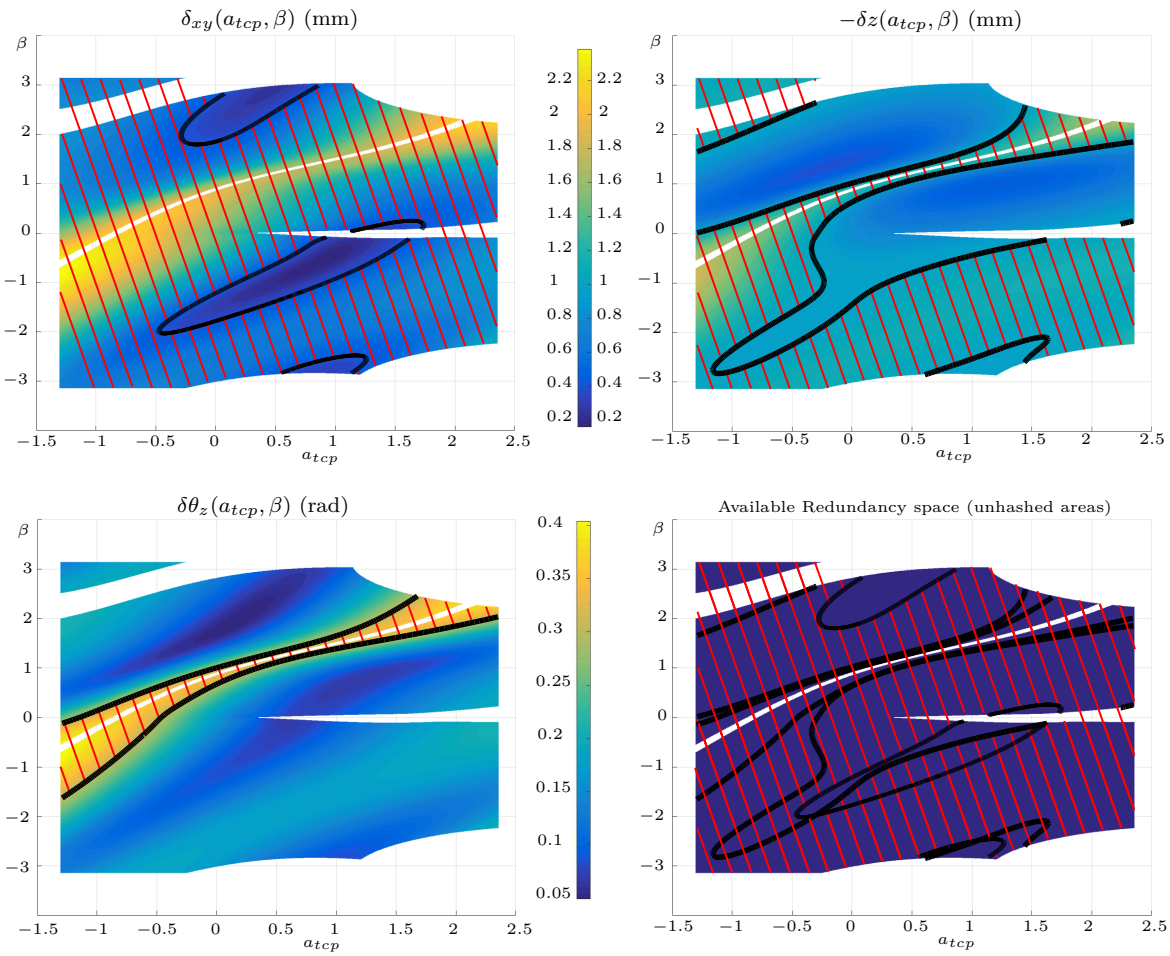


Figure 2.5.2: Various misplacement measures put in perspective for a spatial force ${}^{tcp}\mathbf{f} = \begin{bmatrix}0N & 0N & -100N & 0Nm & 0Nm & -5Nm\end{bmatrix}^\intercal$ applied at the TCP. Drilling plane displacement (top left), displacement along drill axis (top right) and drill axis angular displacement (bottom left) are displayed with regards to the two redundancy parameters β and a_{tcp} for a force. On the example, the tolerances are $\delta_{xy_{max}} = 0.4$ mm, $\delta_{z_{min}} = -1$ mm and $\delta_{z_{max}} = 1$ mm, $\delta_{\theta_{max}} = 0.3°$. A set of redundancy space positions can be selected to comply with tolerances on each misplacement (bottom right).

end effector.

We demonstrated the benefits of using the redundancy space framework in order to find suitable configurations. The main use-case presented was one for which the robot considered had 2 internal motions causing no change in the task fulfilment. For redundancy spaces composed of more dimensions, the representation of the performance criteria cannot be so intuitively done, despite the fact that the performance criteria can still be computed easily for each sampled redundancy space position. Inspite of this, optimisation schemes can easily be implemented to improve the deformational behaviour of redundant systems. The formulation of such an optimisation problem can be found in Section 1.4, with the benefits redundancy space parameterisations provide.

We demonstrated how a constrained satisfaction problem could formalise a redundancy resolution problem, in order to constrain the values of several performance criteria between user-defined boundaries (Fig. 2.5.2).

In the next chapter, the problem of force capacity of serial redundant manipulators will be tackled. Another performance criterion will be defined that can be exploited similarly as the ones related to the deformational behaviour of the robot.

3

Force capacity improvement of serial redundant manipulators

When considering a set of robotic tasks involving physical interaction with the environment, the theoretical knowledge of the full force capacity of the manipulator is a key factor in the design or development of an efficient and economically attractive solution. Carrying its own weight while countering forces may be too much for a robot in certain configurations. Kinematic redundancy with regard to a task allows a robot to perform it in a continuous space of articular configurations; space in which the payload of the robot may vary dramatically. It may be impossible to withstand a physical interaction in some configurations, while it may be easily sustainable in others that bring the end-effector to the same location. This becomes obviously more prevalent for a limited payload robot.

This chapter describes a framework for this kind of operations, in which kinematic redundancy is used to explore the full extent of a force[1] capacity for a given manipulator and task.

[1] In this chapter, the terms "force" and "wrench" may interchangeably refer to 2,3 or 6 dimensional forces depending on the dimension of the problem (planar, spatial) and on whether they may or may not include components of translational forces and/or moments. Their dimensional definition will be explicitly given whenever specifically needed.

Contents

3.1 Force Capacity analysis in the literature

Recent developments in lightweight robotics put versatility on center stage. This versatility is however often hampered by their limited force capacity. This is still an obstacle to have these manipulators populate a larger portion of the industrial robotic scene. The idea behind this chapter comes from our participation to the Airbus Shopfloor Challenge taking place at ICRA 2016 Stockholm (see Fig. 3.1.1), where a frustrating need for a redundancy exploitation to the benefit of force capacity emerged. Years of study on the subject of poly-articulated mechanisms have shown us that the force capacity of a robot strongly depends on its articular configuration [55] [56] [57] [58] [13] [59]. We propose in this chapter to exploit these variations to the manipulators' advantage.

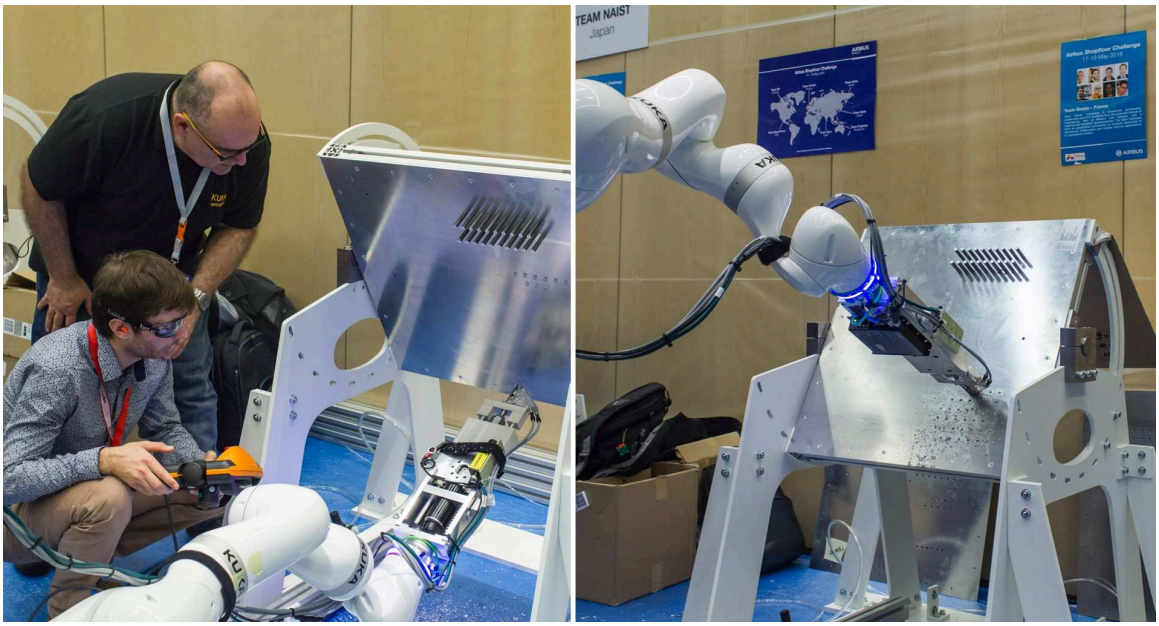


Figure 3.1.1: 7-joints LBR iiwa during the Airbus Shopfloor Challenge, ICRA 2016, Stockholm.

Yoshikawa introduced the concept of force manipulability ellipsoids [55]. These ellipsoids are ways of representing, in a given articular configuration, the force trans-

mission efficiency of serials manipulators. The force ellipsoid is defined as the forces created by the set of all possible torque vectors whose norm is equal to one, when the manipulator is in a given posture. This sphere of articular torques is mapped, thanks to the Jacobian matrix of the manipulator, into Cartesian space to form an ellipsoid of Cartesian forces. Yoshikawa also designed the concept of dynamic manipulability ellipsoids [50], which accounts for the efficiency of a manipulator to produce accelerations of its end-effector. This concept was refined to better take into account the effect of gravity [60] and Chiacchio and Concilio later formulated a version of this ellipsoid for redundant manipulators and non-redundant manipulator in singular configurations [52] [56]. On the downside, ellipsoids only provide an approximate description of the performance of a manipulator [61], for they are derived from a l_2 norm rather that from a l_∞ norm, which prevents them from transforming the exact joint constraints into task space [62].

The force polytopes [57] are another well known tool for describing dexterity. Force polytopes have the ability to accurately describe maximum achievable force capacity of manipulators, because they derive from a l_∞ norm. A force polytope is constructed by mapping the exact joint torque constraints, depicted as a convex polytope in articular space, into Cartesian force space, to form a convex polytope. The force polytope provides the exact Cartesian force capability bounds of the robot in all possible Cartesian forces directions for one given posture. They share with ellipsoids problems of homogeneity that arise when using euclidean metrics mixing angular and translational components, and problems of dependency to scale and units [52]. In the redundant or singular case, the computation of polytopes and ellipsoids requires a generalised inversion or Singular Value Decomposition (SVD) of the Jacobian matrix, as well as a projection on the range space of J^T [56], to remain compatible with the static constraints that were outlined in [63].

Later on, Bolwing and Khatib developped the dynamic capability equations (DCE) [58]. This tool expresses the translational and orientational components of Cartesian speed, acceleration, force capacities altogether in joint torque space and makes these immediately comparable in terms of their torque contribution. To do so, spheres of

minimal Cartesian speed, acceleration and forces are mapped into joint torque space thanks to the named equation and their corresponding joint torque ellipsoids are accumulated, centering one ellipsoid after the other on the surface of the previous one. This accumulation has to be contained within the joint torque polytope for the manipulator to be able to achieve the minimal performance in terms of the Cartesian quantity in or about any direction. The strengths of this tool are its ability to unify velocity, acceleration and force analysis, while dealing consistently with translations and rotations together.

The methodology proposed in this chapter aims at accurately characterising a specific wrench capacity on the full extent of the redundancy space of serial redundant robots to choose a suitable configuration for a given task. The incentive behind this focus on a specific static wrench capacity is that many industrial operations involve specific wrench sustainment and fixed end-effector location. Hence, choosing a suitable destination for this job is very important. Another incentive lives in the need for simplicity and industrial usability of a force capacity criteria. The placement of the base of non-redundant robots and the offline planning (OLP) of redundant robots are often made empirically and without measurable exploitation of their potential benefits, despite the existence of very interesting and global tools, like the ones described before.

The force capacity index (FCI) developed herein is a real-valued index, that exists for an input configuration and an input wrench. This index gives a succinct and quick estimation of the maximum intensity, *i.e.* the multiplier of the input wrench, that wouldn't saturate any actuator of a manipulator, while the latter is already sustaining its own weight.

3.2 Comparison with traditional tools

A comparative study of the FCI with traditional force performance evaluation tools is presented in Table 3.2.1. The ongoing section will complement this table, providing

	Force Polytopes	Force Ellipsoids	Manipulability index	DCE Framework	FCI
Requires Gen. Inv. or SVD	Yes	Yes	Yes	No	No
Output type	2 x 3D polytope	2 x 3D ellipsoid	Scalar	Accumulated N-D ellipsoids	Scalar
Range of study	All Cartesian directions	All Cartesian directions	All Cartesian directions	All Cartesian directions	Focused on a single wrench
Can estimate specific wrench production performance	Yes, for pure force/moment	Yes, for pure force/moment	No	Yes, but at a great cost	Yes
Combines forces/moments in a physically meaningfull manner	No	No	No	Yes	Yes
Independence of scale and unit	No	No	No	Yes	Yes
Handles well singular config. Scales well with #DOFs	No	No	No	Yes	Yes
Analytical expression exists	No	No	No	Yes	Yes

Table 3.2.1: Qualitative comparative study of different posture-dependent tools used for wrench production performance evaluation of serial redundant manipulators.

relevant details about each tool, and detailing how and under which assumptions a result equivalent to the FCI output could be obtained from this list of tools.

Force Polytopes : The use of force polytopes would not suit this particular purpose in most cases, as general wrenches mix forces and moments, which can't be meaningfully combined with this tool. For pure force/moment wrenches however, one could try to look at the intersection of the inquired wrench direction (straight line in the relevant Cartesian space) with the surface of the force polytope, which would provide a result equivalent to the FCI's.

Force Ellipsoid : Force ellipsoids share the same problem of being unable to meaningfully combine forces and moments. Besides, ellipsoids account for force production *efficiency*, not *capacity* (*efficiency* at producing a wrench may be high even though a robot joint is close to a torque limit, thus limiting the robot wrench production *capacity*). Regardless, specific wrench production *efficiency* could be computed for pure force/moment wrenches expressing these wrenches into the singular vectors decomposition associated with the configuration of the system and weighting accordingly with the associated singular values.

Manipulability Index : The manipulability index is encompassing all Cartesian force production *efficiencies* into one single scalar and thus doesn't allow inquiry about a specific wrench production *efficiency*, let alone *capacity*.

DCE Framework : Using the DCE framework for a specific wrench capacity analysis would be irrelevantly-computationally expensive. It would require accumulating, within the torque vector space of the manipulator, an ellipsoid corresponding to a pure force wrench that has the same norm as the one coming from the specific wrench with ellipsoids corresponding to a pure moment wrench that has the same norm as the one coming from the specific wrench. Then, it would require scaling up these ellipsoids until the specific wrench lay on the surface of the robot torque polytope. This final scaling factor would then be equivalent to the FCI output. This comparative study

clearly highlights the need for such an index.

The methodology proposed in this chapter shares with the DCE's its ability to mix Cartesian translational and rotational forces in a physically meaningful manner, and dealing with the problems of dependency on scale and units by focusing on geometric computations in joint torque space. Just like force ellispoids and polytopes, the DCE gives global insights of the force performances of a manipulator, in and about any direction of the Cartesian space. This particular feature is desirable for robot design purpose, or for reactive control applications, where unforeseen physical interactions or reactions may have to be dealt with. This, however, does not count among the explicit focuses of our framework. On the bright side, the FCI behaves very well with high-number-of-DOFs-manipulators, doesn't suffer from singularities, doesn't involve costly computations (such as generalised inversion or SVD), and most importantly has a straightforward formulation and interpretation. These features make it very suitable for redundancy resolution schemes, for easy implementation in autonomous and/or offline programming of redundant robots, in the mechanical design of end-effectors, and possibly in control schemes, although adjustments would have to be done in the latter case to account for dynamic effects. Control schemes won't be exemplified in this chapter, which aims at introducing the FCI and apply its use to quasi-static applications.

3.3 Proposed methodology

3.3.1 The force capacity index λ_{sat}

In the n-dimensional affine space of an n-DOFs manipulator joint torques, let us define the joint torque polytope (see Fig. 3.3.1) for a 3-DOFs manipulator) as the convex n-dimensional polyhedron described by the bounding inequalities: $\forall i \in [\![1, n]\!], \tau_{i,l} \leqslant \tau_i \leqslant \tau_{i,u}$, which expresses the fact that the torque of joint i, τ_i, lies within the torque lower bound $\tau_{i,l}$ and the torque upper bound $\tau_{i,u}$ segment.

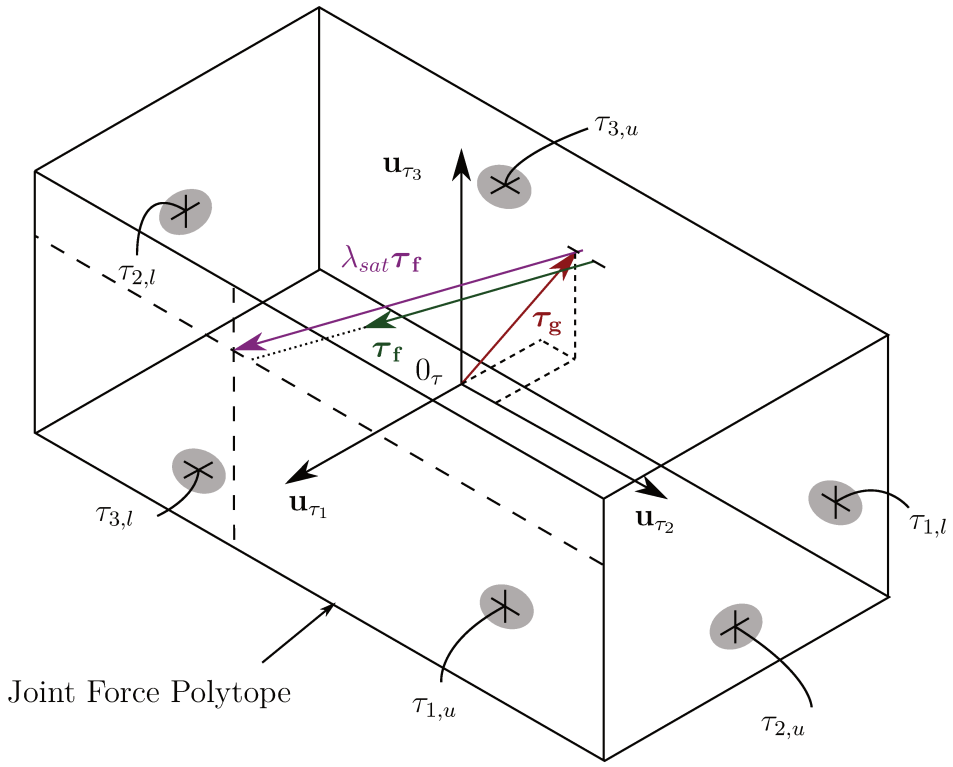


Figure 3.3.1: Joint force polytope for a 3-DOFs manipulator with torque limits $\tau_{i,l}$ and $\tau_{i,u}$ for $i \in [\![1, 3]\!]$

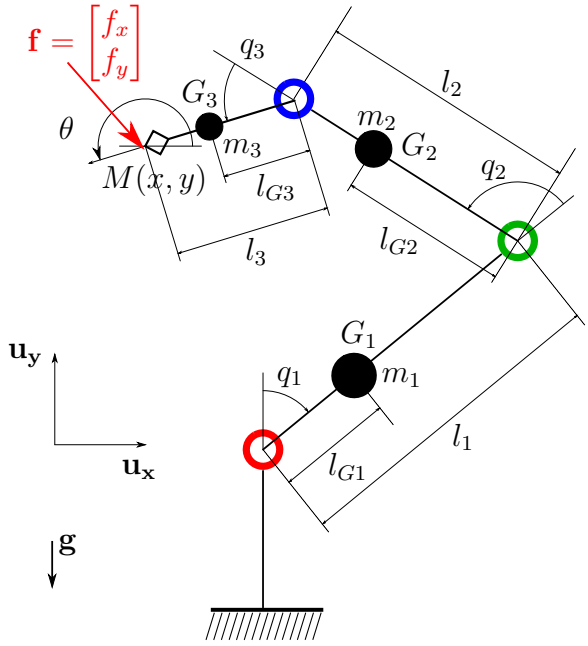


Figure 3.3.2: A 3-DOFs planar manipulator settings : $l_1 = 0.7\,\mathrm{m}$, $l_2 = 0.6\,\mathrm{m}$, $l_3 = 0.4\,\mathrm{m}$, $m_1 = 1.5\,\mathrm{kg}$, $m_2 = 1\,\mathrm{kg}$, $m_3 = 0.5\,\mathrm{kg}$, $l_{G1} = 0.5l_1$, $l_{G2} = 0.5l_2$, $l_{G3} = 0.66l_3$, $\tau_{1,u} = -\tau_{1,l} = 20\,\mathrm{N\,m}$, $\tau_{2,u} = -\tau_{2,l} = 9\,\mathrm{N\,m}$, $\tau_{3,u} = -\tau_{3,l} = 6\,\mathrm{N\,m}$. *Note:* Two sets of solutions exist for this planar arm as link 1 and link 2 may also be symmetrically placed on the other side of the joint 1-joint 3 axis. However, only one of these sets will be explored, for the sake of brevity.

Let $\mathbf{f}$ be the wrench, expressed at the TCP[2], that the manipulator has to sustain at minimum and let $\mathbf{q}$ be the articular configuration in which to compute the wrench capacity. The gravity torque vector $\boldsymbol{\tau}_\mathbf{g}(\mathbf{q})$, i.e. the vector containing the torque that each actuator of the manipulator will have to produce to counter the effect of weight only, can then be computed thanks to the mass data of the manipulator. Then, the torque vector $\boldsymbol{\tau}_{\lambda\mathbf{f}}$, needed to sustain a lone spatial force $\lambda\mathbf{f}, \lambda \in \mathbb{R}^+$ in the configuration $\mathbf{q}$ can be computed thanks to the kinetostatic equation $\boldsymbol{\tau}_{\lambda\mathbf{f}} = \lambda J(\mathbf{q})^T\mathbf{f} = \lambda\boldsymbol{\tau}_\mathbf{f}$. We will refer to λ as the intensity of the spatial force. During the operation, both $\boldsymbol{\tau}_\mathbf{g}$ and $\boldsymbol{\tau}_{\lambda\mathbf{f}}$, with $\lambda = 1$, have to be produced at the same time, and we want to know the maximum intensity of the wrench that can be sustained by the manip-

[2]The TCP, or Tool Center Point, refers to a frame, which is statically attached to the extremal link of the manipulator. This frame is generally placed and oriented at an operating location of the end-effector.

ulator. Determining the wrench capacity comes down to computing the intensity $\lambda_{sat}(\mathbf{q})$ multiplying the wrench $\mathbf{f}$ below which no actuator saturates, and from which at least one actuator saturates. This geometrically translates into finding λ_{sat} so that $\boldsymbol{\tau}(\mathbf{q}) = \boldsymbol{\tau}_{\mathbf{g}}(\mathbf{q}) + \lambda_{sat}\boldsymbol{\tau}_{\mathbf{f}}$ lies on the surface of the joint torque polytope, as can be seen on Fig. 3.3.1. Practically, it is therefore interesting to look into the saturating intensity of each actuator, *i.e.* the minimal intensity $\lambda_i(\mathbf{q}) \geqslant 0, i \in [\![1, n]\!]$ of the wrench $\mathbf{f}$ that would overwhelm actuator i, without consideration for the other actuators. This intensity of the wrench produces a torque $\tau_i(\mathbf{q})$ at actuator i either equal to $\tau_{i,l}$ or to $\tau_{i,u}$. The minimum of the n saturating intensities exactly corresponds to $\lambda_{sat}(\mathbf{q})$.

$$\forall i \in [\![1, n]\!], \lambda_i = \begin{cases} \frac{\tau_{u,i} - \tau_{\mathbf{g},i}}{\tau_{\mathbf{f},i}}, \text{ if } \tau_{\mathbf{f},i} > 0 \\ \frac{\tau_{l,i} - \tau_{\mathbf{g},i}}{\tau_{\mathbf{f},i}}, \text{ if } \tau_{\mathbf{f},i} < 0 \\ \infty, \text{ if } \tau_{\mathbf{f},i} = 0 \end{cases}$$

$$\lambda_{sat} = min((\lambda_i)_{i \in [\![1,n]\!]})$$

3.3.2 A simple example : 3-DOFs planar robot for 2-D positioning task

Let us consider the planar manipulator with three serial revolute joints depicted (for three different configurations) in Fig. 3.3.2 for the 2-dimensional planar positioning task while the robot is applying a linear 2-dimensional force $\mathbf{f}$ at its TCP. The robot is kinematically redundant of order one for the positioning task and therefore has a one dimensional space of articular configurations that allows for the position of its TCP to match the task requirements. The initial step is to find an intuitive parameterisation of the redundancy space. In our situation, a solution could be to take the third joint coordinate of the manipulator, which is, according to the formulation of [64], a monotonic type joint. Indeed, regardless of the goal position occupied by the TCP, fixing the value of this joint angle enables to provide at most one finite set of solutions to the inverse geometric model. Another solution for parameterising the redundancy

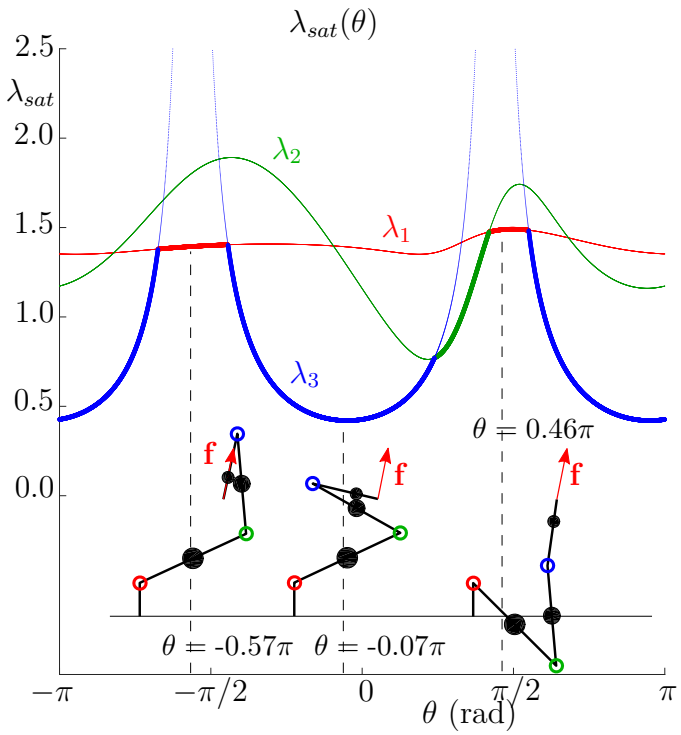


Figure 3.3.3: Joint saturating intensities λ_1, λ_2 and λ_3 (thin lines) and saturating intensity λ_{sat} (thick line) of the 3-DOFs planar robot described in figure 3.3.2 for $\theta \in [-\pi, \pi]$. Task destination is $M(0.5\text{ (m)}, 0.5\text{ (m)})$ and output force is $\mathbf{f} = [5.5\text{ (N)}, 27.5\text{ (N)}]^T$ (everything is expressed in the origin reference frame). Some enlightening postures of this manipulator are shown below the curves.

space of the problem is to use the angle θ which corresponds to the angle between the horizontal and the pointing direction of the TCP. Both parameterisations would work, but we will favour the latter, because a closed-form expression for the inverse geometry problem is more trivially found this way.

Fig. 3.3.3 is a representation of the saturating intensity of the three actuators of the manipulator over a significant sampling of $\boldsymbol{\alpha} = \theta \in [-\pi, \pi]$. For each sampled θ, the value of λ_{sat} is the minimum value of the saturating intensities λ_1, λ_2 and λ_3. The wrench capacity is shown to greatly depend on the value of the redundancy parameter θ. In this simulated example, λ_{sat} varies everywhere between 0.43 and 1.50. The wrench $\mathbf{f}$ is not feasible when $\lambda_{sat} < 1$. Drawing a horizontal line at this $\lambda_{sat} = 1$ highlights the admissible ranges of θ for the current task and spatial force, which

leaves room for further improvements from a redundancy resolution perspective. In the example, we can see that two spans of values for θ are available for the manipulator to sustain the wrench $\mathbf{f}$ with an intensity $\lambda_{sat} \geqslant 1$.

Links length and mass, positions of centres of gravity and TCPs, as well as torques limits also have a strong and varying influence on the shape of the saturating intensities. The FCI can also be an interesting criterion to use during robot or end-effector mechanical design, as is exemplified at the end of the chapter.

3.4 Application of the method to the LBR iiwa

Let us apply this method on an existing robot, the KUKA LBR iiwa, for drilling operations. The drilling task is a fully constrained task with a free rotation about the drill axis. Therefore, the it only constrains five DOFs, which leaves us with a 2-redundant-DOFs manipulator. In these condition, we choose to take the rotation about the drill (denoted a_{tcp}) and the elbow angle (denoted β) as parameters for the redundancy parameterization of our problem.

Fig. 3.4.1 shows the output of the force capacity evolution over a meaningful sampling of this 2-dimensional redundancy space. In this simulation, λ_{sat} has very sharp variations, with a potential gain that approximates +290% in the force capacity index. The robot postures associated to three relevant positions in redundancy space are also displayed to give further insight on how the redundancy position influences the general posture of the manipulator while complying with the task, and what makes the robot strong against the spatial force depicted by the red arrow. An intuitive result is that the robot seems stronger when its weakest joints are lightly solicited (in the LBR iiwa, the torque limits range from ±320 Nm for the first two base joints to ±40 Nm for the last two joints); *e.g.* only when joint 6 aligns its rotation axis with the direction of the force does the robot get sharply stronger[3]. On the other hand, joint 6 seems to be highly solicited, with its axis nearly orthogonal

[3] Joint 7 solicitation does not change much over the entire redundancy space, and the joint is only very lightly burdened whatsoever.

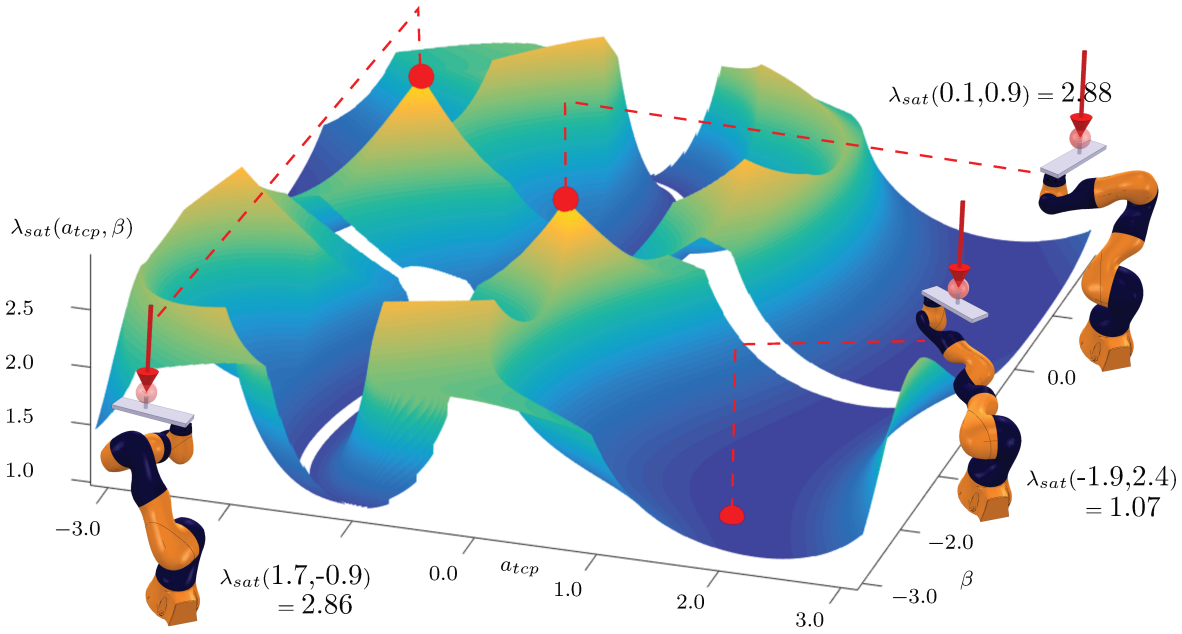


Figure 3.4.1: Force saturating intensity of a KUKA LBR iiwa R820, mounted with a 5.0 kg end-effector, for a spatial force depicted by the red arrow (itself pointing at the TCP and along its z_{tcp} component) with translational component $\boldsymbol{\mathcal{R}}(EE \rightarrow ext) = [0\ (\text{N}), 0\ (\text{N}), -200\ (\text{N})]_{\mathcal{B}_{tcp}}$ and moment $\boldsymbol{\mathcal{M}}_{O_{tcp}}(EE \rightarrow ext) = [0\ (\text{Nm}), 0\ (\text{Nm}), 0\ (\text{Nm})]_{\mathcal{B}_{tcp}}$ (expressed at the TCP origin O_{tcp}, in its associated geometric basis $\mathcal{B}_{tcp}$). The 5-DOFs task is defined by the position : $[x, y, z] = [0.580\ (\text{m}), 0\ (\text{m}), 0.740\ (\text{m})]_{\mathcal{B}_0}$ and orientation (with XYZ-euler formalism) : $[\alpha_x, \beta_y, \gamma_z] = [0\ (\text{rad}), 0\ (\text{rad}), -0.8153 + a_{tcp}\ (\text{rad})]_{\mathcal{B}_0}$. The FCI value in redundancy space zones which are out of joint movement limits or otherwise unreachable don't appear on the figure (blank gaps).

to the force direction, in the weaker posture shown in the bottom right of Fig. 3.4.1.

A practical use case which consists in drilling a set of y-direction-aligned holes from under a surface with a LBR iiwa (14kg payload) was simulated and experimentally tested to illustrate a way the FCI could be use for discrete path planning and end-effector mechanical design purposes. Let us suppose that the unfinished design of the end-effector leaves us with two degrees of freedom on the position and orientation of the drill, as illustrated on Fig. 3.4.2. The drill rotation and feed motions are exercised by internal mechanisms of the end-effector. The wrench associated to the

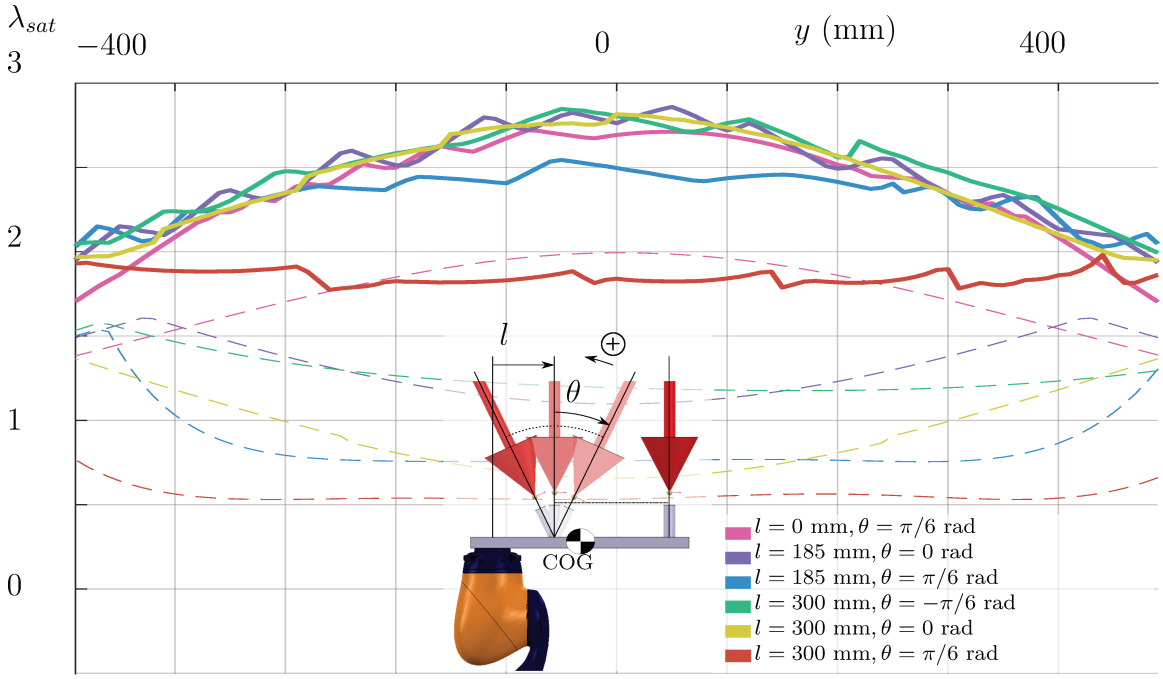


Figure 3.4.2: Evolutions of λ_{sat} along the holes span for six different end-effector designs in the *standard* (dashed lines) and *redundant* (thick lines) scenarios accompanied by an illustration of the end-effector design DOFs. The curves show how optimising the posture of the robot may drastically enhance the force capacity of the system, and how much a change of end-effector design may change this capacity.

drilling operation is the one used for Fig. 3.4.1, which thus corresponds to the nominal hole operation task ($y = 0$ mm). We will consider a first scenario, referred as the *standard* scenario, where the orientation of the end-effector is fully constrained for each hole position (thus leading to 6 geometric constraints for the task) and where the position of the robot elbow (swivel angle) is set to a constant $\beta = 0°$. One may note that this first scenario is very likely to happen on real industrial applications, given that the default API of the robot currently doesn't allow easy exploitation its intrinsic redundancy. We will also consider a second scenario, referred as the *redundant* scenario, where the robot has the same 2-dimensional redundancy space as the one used for Fig. 3.4.1, for each hole position.

The simulations on Fig. 3.4.2 display, for 6 different designs of end-effectors, the

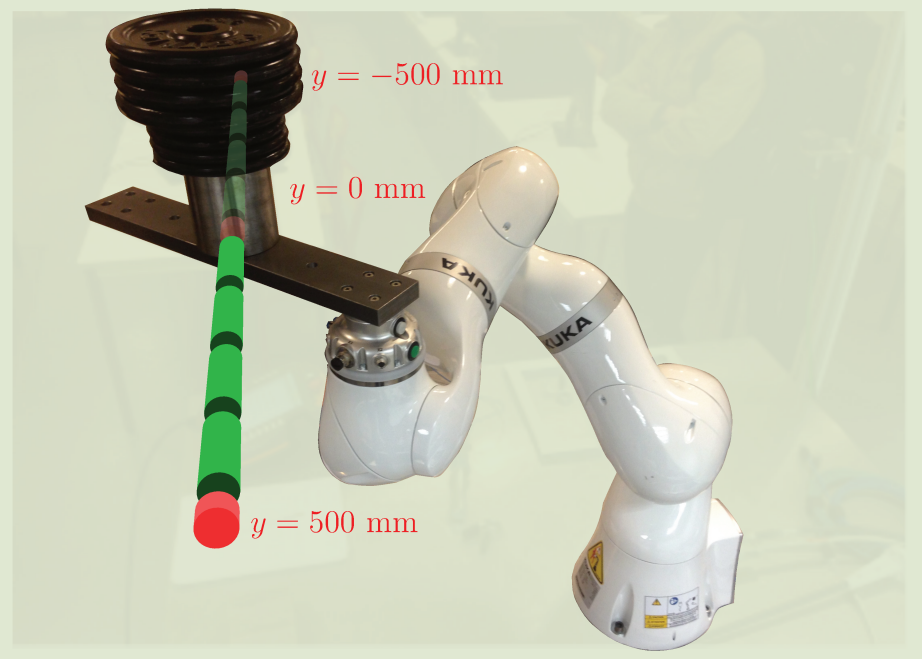


Figure 3.4.3: Use case experimentation and illustration of the holes span. KUKA LBR iiwa easily withstanding 5 kg end-effector and 22 kg load.

variations of the FCI along the range of the hole plausible positions. In dashed lines are simulated the saturating intensities of the *standard* scenario. In thick lines are simulated the saturating intensities of the *redundant* scenario. Each simulated end-effector design corresponds to a different color on the graph of Fig. 3.4.2. It is worth noticing that the mechanical design of the end-effector has consequences on the manipulator force capacity that one would expect in the *standard* scenario (namely that the robot is getting weaker with regard to the wrench as l and θ increase), but that this behaviour becomes less obvious when full exploitation of redundancy is performed. Redundancy helps balancing torque solicitations on joints and sometimes gives astonishing results. Fig. 3.4.3 shows a practical test of this use case, for a 5 kg end-effector subject to a 22 kg payload, for $y = 0$ mm. The configuration used for this test is the same as the one displayed in the top right of Fig. 3.4.1, and proved to be uncharacteristically strong for this relatively low payload robot.

On the practical aspect of computations, the non convexity, non-smoothness and

sometimes non-continuity of λ_{sat} function, which comes from taking the minimum of the (non-convex) saturating intensities[4], from the non-reachability of some redundancy space positions and from joint limits, makes it a challenging problem for traditional optimisation approaches taking λ_{sat} as cost (or constraint) function, without guarantee of finding a global maximum. Local maxima would also be difficult to find for traditional optimisation techniques because they often lie on, or close to the intersections of different saturating intensities, i.e. areas where λ_{sat} function is not differentiable.

Low dimensional problems like the ones shown in this chapter can be fully and efficiently solved by sampling the redundancy space of the manipulator and computing λ_{sat} for each sample. The sampling approach allows for a very thorough understanding of the possibilities redundancy offers. For higher dimensional problems (i.e. problems where the redundancy space has many degrees of freedom), it may be advisable to use optimisation techniques such as genetic algorithms, which are more suited to these kinds of functions than algorithms traditionally used in constrained non-linear optimisation (interior point, SQP, active set, trust region reflective, etc...) although they don't guarantee the repeatability of the result. Their use was experimented with success on 1-, 2- and 4-DOFs redundancy space, with good results (the local extrema shown on Fig. 3.4.1 are results of this genetic algorithm approach).

3.5 Intermediary Conclusion

A real-valued index measuring the capacity extent of a serial manipulator, set in a given configuration, to produce a specific spatial force was described in this chapter. A framework, based on a parameterisation of the self-motion of redundant manipulators, was then presented to help explore their redundancy space in search for strong values of the force capacity index.

Unlike polytopes, manipulability ellipsoids and DCE-linked tools, which focus on the overall force (and other physical quantities) capabilities in one given configuration, this method focuses on one particular force capacity over all the possible configura-

[4]For examples of non-smooth and non-convex saturating intensities, see Fig. 3.3.3.

tions offered by kinematic redundancy. The authors find this tool suitable for direct use, with the purpose of simplifying redundancy resolution and exploitation.

The relevance of the proposed approach was illustrated with a practical use case involving a discrete path planning and mechanical design problem. The use case demonstrated the advantage of exploiting the available redundancies of this system while demonstrating the simplicity of use of the FCI. The presented index was shown to have a strong dependency on the configuration used by the manipulator, which demonstrates its relevance for redundant robotic operations involving interactions.

Quite remarkably, KUKA LBR iiwa 14 R820 is shown to have force capabilities which extended well beyond what the worst-case-14kg-equivalent-weight announced by the manufacturer could suggest (on the simulation of Fig. 3.4.1, more than 570 N can be withstood at the TCP).

Although force production may be extended well beyond what one could expect from a robot such as the LBR iiwa, it may be important, for industrial operations, to maintain a level of accuracy by limiting gear elastic deformations. The criteria presented in Chapter 2 may be used in parallel to the force capacity index in order to ensure a sufficient force capacity while limiting the deformational behaviour of the robot operating point.

4

Motion Planning in Dynamic Environment Using Receding Horizon Dynamic RoadMaps

Performing motion planning in an industrial, human populated environment is a very challenging problem. The industrial context imposes robustness and efficiency, while the presence of human imposes safe trajectories and creates dynamic obstacles.

In the literature, a lot of focus is made on fast planning rates or path optimality for a given static environment. A broadly accepted concept to cope with moving obstacles is to use reactive control but this strategy is prone to local minima and deadlocks.

In this chapter, a strategy that uses an anticipation of the obstacle motions is presented to create safe and robust trajectories. A method called Receding Horizon Dynamic RoadMaps (RH-DRM) is presented that allows efficient and safe trajectories in changing environments. This method, which classifies into the Sampling Based (SB) approaches, uses the concept of Dynamic RoadMaps (DRM) and extends it to obstacles anticipated trajectories. A novel graph search method within a time changing graph is presented to solve the problem of the earliest arrival journey within an anticipated roadmap topology.

The chapter first introduces the broad field of motion planning. Then, it focuses on approaches inspired from the well-known PRM method and introduces core tools used by these strategies. Eventually, the RH-DRM method is presented.

Contents

4.1 Context

The problem of managing collaborative and mobile systems within a dynamic a cluttered environment is a complex one. More specifically, managing a flock of hyper redundant, inaccurate, mechanically flexible and power limited iiwa-type robot arms mounted on KMR-type robotic platforms, in a human-populated aircraft production shop floor, to have them safely and autonomously perform assembly, drilling, or measurement operations is a real technological challenge. Managing these systems in this environment involves several steps.

On a high level of abstraction, it involves dynamically assigning "jobs" to the available systems. This operation is often referred to as *task planning*. A "job" for a robot generally consists of performing a unique type of process in a set of locations. We will assume that these locations are organised geographically and can be reached from a unique mobile platform location. In this thesis, the task assignment is assumed to be an entry point of our problem.

From this job allocation, one system can pick a suitable geometric destination for its mobile platform that allows the robot arm to suitably reach the set of locations associated to the job. This step involves performing a redundancy resolution scheme. Tools to perform this step were introduced in the previous chapters.

Then, a system needs to safely travel to its chosen destination. Part of the environment is well known and static. For instance, the fuselage geometry is known in advance. Aside from that, the environment is occupied by other robots and human beings that move about the place. One part of this problem is knowing the environment by constantly sensing it. The other part, which is the subject of this chapter, is planning a suitable motion within this dynamic and cluttered environment.

Figure 4.1.1: Mobile robot having to plan a motion in a human populated environment.

4.2 Motion planner specifications

In this chapter, we will consider the problem of planning the motions of a robot, which may be a serial arm, a mobile platform, or an hybrid system that couples them both. We will consider no differential constraints, as both robots can be considered holonomic. The system, whatever it may be, performs operations in a place shared by robots and human workers. It comprises the boundaries of the place, like the walls, floor and other fixed obstacles. It will therefore be referred to as the *static environment*. Human workers and other robots will be referred to as the *dynamic environment* of the system. The dynamic environment behaviour is not known in advance. Therefore, the motion planning strategy must adapt to them autonomously.

The objectives that we want to meet for this motion planning solution are several fold:

- **Motion safety :** The motion planning solution has to be safe. The system should avoid collisions with its environment while moving about, if it is avoidable. For the sake of generality, the system is not assumed to be faster than the other moving agents. It will have to cope with obstacles that may have higher dynamics than itself.

- **Motion efficiency :** The method has to deliver an efficient solution by choosing an early arrival route to the destination.

- **Motion induction and intrusiveness :** The path taken by the system must adapt to the environment and should avoid disrupting the dynamic environment's flow. We believe it to be an important feature for improving human acceptability.

- **Geometric detailing :** A lot of existing algorithms make strong assumptions and simplifications on the shape of the system and obstacles. These assumptions generally aim at decreasing the computational load to an acceptable level. These assumptions are sometimes so profoundly rooted in the methods that they cannot be relaxed without modifying completely the inner working of the algorithms. These simplifications are often limiting in the context of a cluttered environment and may prevent nimble trajectories from being found. The method developed for our problem must provide the means to choose the detailing level of the geometric shapes that compose the environment and system. This specification will allow the efficiency and result quality of the algorithm to improve with the computational power increasing trend.

- **Computational cost :** The method has to be able to cope with sudden changes in the environment while remaining safe. It should therefore be light and fast, programmatically speaking.

- **Adaptability to multi-robot motion planning :** Ideally, the solution should also be adaptable to a multi-robots motion planning problem. It should scale well with the number of robots.

- **Industrialisation ease :** The method is to be robust and deterministic. The same consequences should arise for two situations with the same problem inputs. Additionally, the inner workings and decision process of the algorithm must be visualisable and understandable. Method transparency is to be sought if the solution is going to be applicable in an industrial context.

4.3 An introduction to some established motion planning schemes

4.3.1 History and definitions

The first motion planning problems were formulated in the mid 60's, but motion planning problems really started drawing researchers attention in 1979, with [65]. Motion planning has since grown to become one of the most studied field in robotics and automation [66–71]. It has various applications in other fields, including computational biology with protein folding algorithms [72], video games or movies with computer animation [73, 74]. Historically, motion planning is often referred to as the piano mover's problem [75]. The goal of motion planning is, for a given system, to produce a collision-free motion from a start to a goal configuration while being among a collection of obstacles. The pair of start and goal configurations is called a *query* in the motion planning vocabulary.

The system and obstacles envelope geometries can be described in a 2D or 3D space called the workspace $\boldsymbol{\mathcal{W}}$ (which can be $\mathbb{R}^2$ or $\mathbb{R}^3$). The motion, however, is conveniently described as a path in (possibly higher-dimensional) configuration space $\boldsymbol{\mathcal{C}}$. The configuration space of a robot is a space that uniquely -and unambiguously- describes the posture of a robot. Articular positions are generally used as the components of the vectors of this space. Using configuration spaces is a useful way of abstracting planning problems in a unified way [76, 77], because articulated robots or robots with complex geometric shapes are represented by a single point $\mathbf{q} \in \boldsymbol{\mathcal{C}}$. The

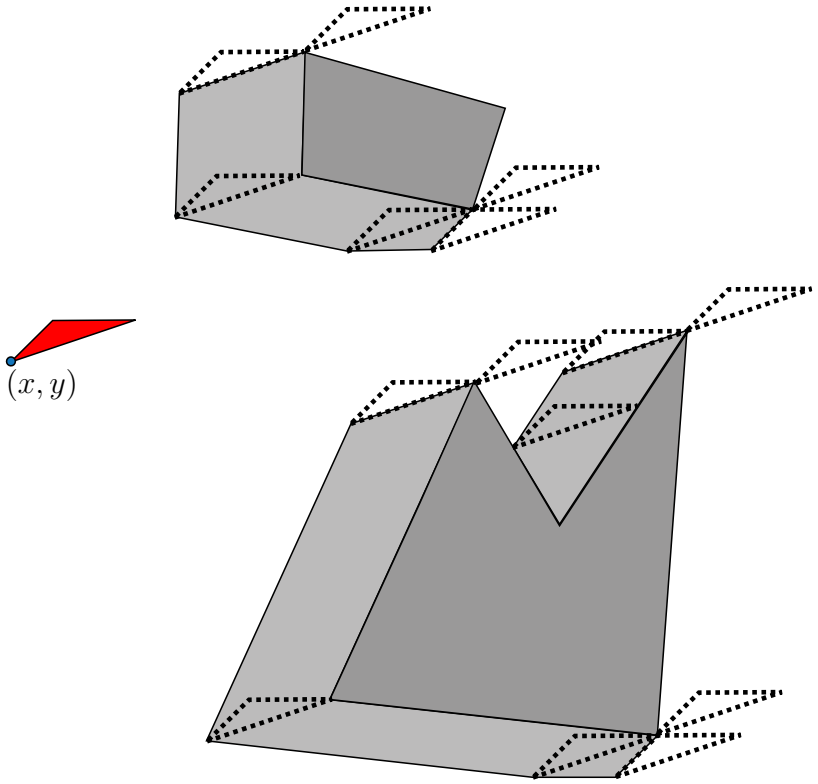


Figure 4.3.1: Definition and construction of $\mathcal{C}_{free}$ for a triangular robot that can move in the plane without rotating. The configuration of the robot is defined as the position of its left bottom vertex : $\mathbf{q} = (x, y)$. Obstacle region $\mathcal{O}$ is displayed in dark grey. Configuration forbidden space $\mathcal{C}_{obs}$ is the union of all shades of grey regions. All non-grey regions are the configuration free space $\mathcal{C}_{free}$.

set of configurations in which the envelope[1] of the robot $\boldsymbol{\mathcal{A}}(\mathbf{q}) \subset \boldsymbol{\mathcal{W}}$ is not colliding with the obstacle region $\boldsymbol{\mathcal{O}} \subset \boldsymbol{\mathcal{W}}$ is called the free configuration space $\boldsymbol{\mathcal{C}}_{free}$ (*i.e.* $\boldsymbol{\mathcal{C}}_{free} = \{\mathbf{q} \in \boldsymbol{\mathcal{C}} \mid \boldsymbol{\mathcal{A}}(\mathbf{q}) \bigcap \boldsymbol{\mathcal{O}} = \emptyset\}$). The free configuration space is illustrated for a simple example in Fig. 4.3.1.The pair composed of start and goal configurations is called the *query* of the motion planning problem.

Given $\boldsymbol{\mathcal{W}}$, $\boldsymbol{\mathcal{O}}$ and $\boldsymbol{\mathcal{A}}$, the motion planning problem can be defined for a query

[1]The envelope is the non convex volume that exactly encompasses the robot for a configuration $\mathbf{q}$.

$(\mathbf{q}_I, \mathbf{q}_G)$, as the search for a continuous path $\boldsymbol{\pi} : [0, t_f] \rightarrow \boldsymbol{\mathcal{C}}_{free}$, such that $\boldsymbol{\pi}(0) = \mathbf{q}_I$ and $\boldsymbol{\pi}(t_f) = \mathbf{q}_G$, where $t_f \geq 0$ is the time needed to reach the destination.

Although the formulation of the motion planning problem is simple, the problem has been proved to be computationally demanding, even for simple systems [75, 78].

Despite a large number of reviews on the field [66–71, 79–82], a taxonomy of motion planning algorithms remains unclear. All the same, some techniques seem to stand out from the lot. The following paragraphs will briefly introduce some of them.

4.3.2 Combinatorial Roadmaps

Combinatorial roadmaps, also referred to as computational geometric algorithms [83] and exact roadmaps, is a discipline that started in the late 60's [84]. It was initially created to solve problems in $\mathbb{R}^2$, with limited possibilities of extension to higher-dimensional spaces. They generally require the full knowledge of the obstacle region.

Roadmaps based on Voronoï diagrams [85–87], also called maximum clearance roadmaps [88] or retraction method [89], are one of these techniques (see Fig. 4.3.2). These algorithms divide free configuration space into regions that have the property of being closer to a single obstacle. The boundaries of these regions have the property of being equally distant to pairs of obstacles, and thus define the maximum clearance roadmaps. Motion planning algorithms fully based on Voronoï roadmaps are complete algorithms. They report in finite time a solution if it exists and are able to report if a query is not feasible.

Visibility graph [83, 84, 90–92], also called shortest path roadmaps, create roadmaps that link all pairs of the polygonal obstacles vertices if they are visible to one another (see Fig. 4.3.3). Paths will actually touch the obstacles, and thus provide the shortest possible solutions.

Cell-based decomposition roadmaps [83, 93–96] use vertical cell decomposition (plane sweep principle) to divide the free configuration space into triangles and trapezoids (see Fig. 4.3.4). Collision free paths can then be computed by moving

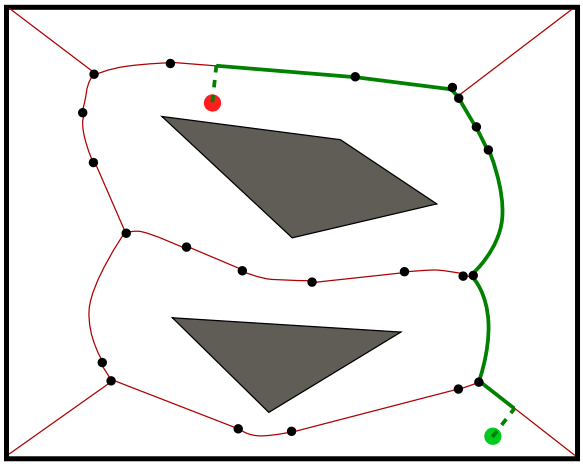


Figure 4.3.2: Voronoi diagram based roadmap.

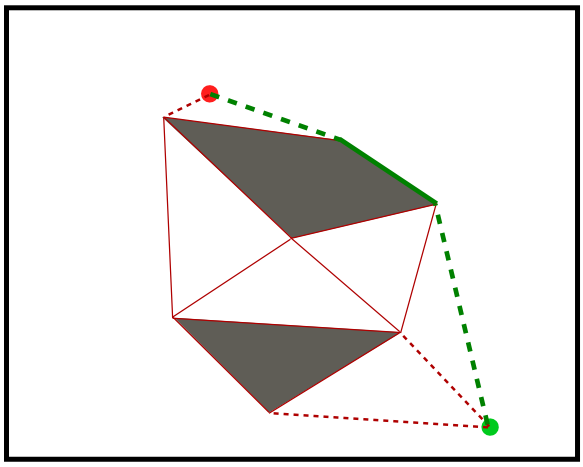


Figure 4.3.3: Visibility graphs, also called shortest path roadmaps.

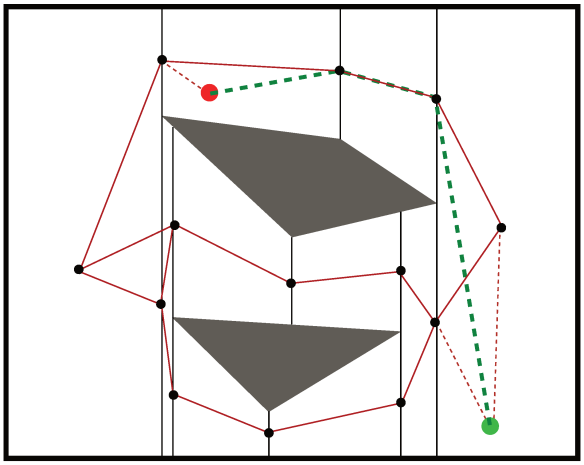


Figure 4.3.4: Vertical cell decomposition roadmaps, based on plane sweep techniques.

from one polygon edges middle point to another.

After computing the roadmap, combinatorial roadmap algorithms use various techniques to insert the query within the roadmaps. After inserting the query, a graph search algorithm such as Dijkstraa or A* is used to find the shortest path.

4.3.3 Artificial Potential Fields

The *Artificial Potential Fields* (APF) approach (Fig. 4.3.5), which was introduced by Khatib in [18], is one of the first successfully motion planning schemes that worked for manipulators and mobile robots. It is a local approach that is very close to control. An artificial force, driving the displacements of a system is computed for a system configuration. This artificial force is defined as the anti-gradient of an artificial potential energy function. This function is the sum of an attractive potential that drives the system towards the goal, and a repulsive potential that drives the system away from obstacles [69].

A simple gradient descent procedure will steer the system towards its destination. Advantages of using this method are numerous. The simplicity of the scheme and its low computational cost make it an appealing solution. It creates smooth and rather predictable trajectories and is well adapted to local and reactive planning. However,

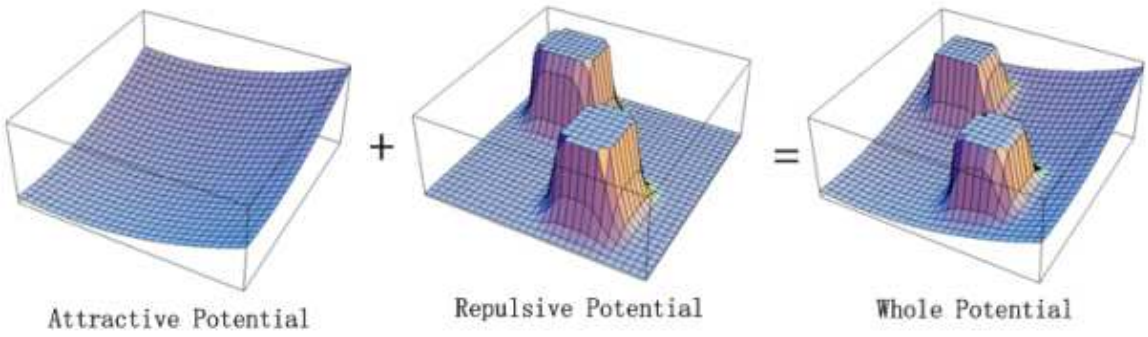


Figure 4.3.5: Artificial Potential Fields

when looking at the global path planning, this greedy approach is not always suitable. The path taken is generated from the local shape of the potential field. In narrow passages, the planning may perform poorly [97]. Moreover, the local feature may easily lead the system into local minima of the potential function without further hope of reaching the destination. Additionally, from a global path perspective, the solution can hardly be optimal. The idea of repulsive APF was introduced in this thesis in the context of local optimisation-based velocity level redundancy resolution (Section 1.2.3) approaches, and can be used in the framework of the task priority approach [98] (Section 1.2.5).

4.3.4 Mathematical programming methods

Mathematical programming methods deal with motion planning using a constrained optimisation method. In this scheme, static and dynamic obstacles avoidance is modelled by inequality constraints on the shortest distance to obstacles. The objective function may be based on the time, the distance and/or the energy required for the motion. An interesting application of this method is the distributed receding horizon approach described in [99, 100]. The method is suited for a flock of robots in a dynamic environment. In this method, at regular time intervals, the planning of each robot is performed and then shared amongst the other members of the flock. Each robot of the flock then modifies its optimal route to adapt to the others trajectory.

4.3.5 Heuristic approaches

The methods mentioned before have the main disadvantage of being "ungeneralisable" to higher dimensional problems or falling in local minima. Heuristics approaches, such as simulated annealing [101–103] have been combined with potential fields to remove the problem of local minima by driving the robot out of them.

Particle swarm optimisation [104–106] and genetic algorithms have also been employed in the motion planning problem [107–110]. In the latter, the main idea is the creation of chromosomes representing the path taken by the robot. Functions computing the distance to collision are used to test the suitability of chromosomes for obstacle avoidance and a suitable cost function can be used to shorten and smooth up the path.

The concept of stigmergy is also relevant for motion planning. Stigmergy is an indirect communication process used by termites and other animals living in colonies. Wandering termites drop pheromones along their way to indicate interesting places or dangerous ones to their congeners. The cumulated experience of many termites may provide accurate information about the environment without requiring any explicit communication. Without physically marking the environment, the same principle was used in [111–114] to guide flocks of mobile robots. A real life application of this technique is done with Google maps application (Fig. 4.3.6), which monitors in real time the traffic density thanks to the GPS signals of their users.

Fuzzy logic has also been used in motion planning schemes [115, 116], in association with neural networks [117, 118], genetic algorithms [119], or potential fields. Other methods include colony optimisation, wavelets or tabu search. A more thorough description and review of these methods can be found in [80].

Despite being rather efficient methods, heuristic approaches have no guarantee of finding a solution if it exists. The main disadvantage of these methods is perhaps the opacity of their inner workings, which could render them difficult to analyse or debug

Figure 4.3.6: The stigmergy principle used for navigation purpose (Map data ©2018 Google).

within an industrial implementation.

4.3.6 Sampling-based methods

Very popular schemes nowadays for motion planning are the sampling based methods (SBM). A comprehensive and recent survey of these approaches can be found in [82]. The two most popular and fundamental schemes deriving from SBM are certainly the *Rapidly-exploring Random Trees* (RRT) [120] and the *Probabilistic Roadmaps* (PRM) [121]. Other very well known planners based on sampling based approaches include the Rapidly-exploring Random Graph (RRG) [122] and Fast Marching Tree (FMT) [123]. The main idea behind sampling based methods is the discretisation of the obstacle free configuration space of robots. No representation of the obstacle configuration space is needed. This representation is a heavy computational burden that generally prevents exact roadmap methods (see Section 4.3.2) from being used in high dimensional configuration space. These methods generally produce a graph structure whose nodes are collision free configurations and whose edges are collision free linear trajectories between configurations. A basic need for sampling based methods is the

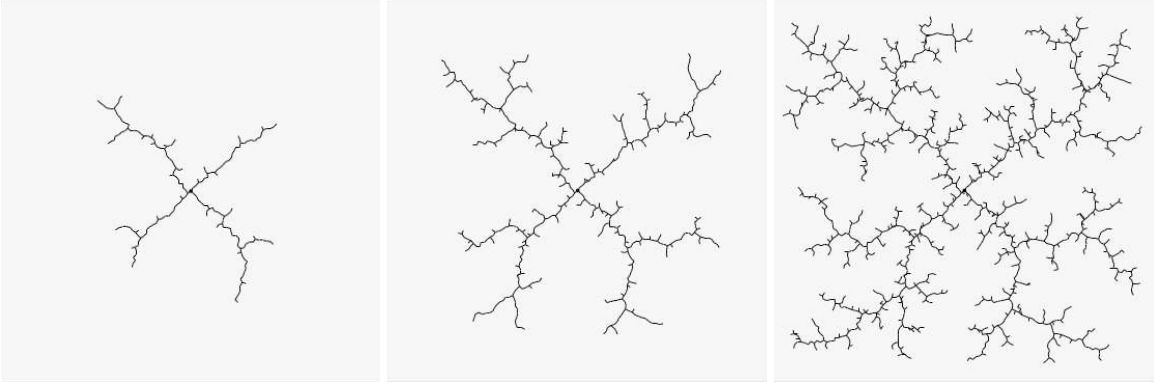


Figure 4.3.7: Original version of RRT algorithm. Expansion is driven towards large Voronoï regions. Credits to Steven Lavalle, in [120], for the illustration showing the expansion of Rapidly-exploring Random Trees.

possibility to perform collision tests very quickly. These tests are performed once for each sampled configuration, and performed repeatedly along the paths that link them together in the graph.

Depending on the method, a graph can be constructed every time a query is issued (single query algorithms), or it can be computed once and then used over and over for all queries to come (multiple-query algorithms).

Rapidly-exploring Random Tree (RRT) : The RRT algorithm is primarily aimed at single query applications. The graph constructed in the original version of the RRT method is rooted at the starting configuration and grows in a somewhat biased manner towards the destination while expanding towards large Voronoï regions (see Fig. 4.3.7). The tree construction process lasts until a configuration situated in the goal region is added to the graph. No tree refinement is performed afterwards, which often leads to highly sub-optimal paths. The absence of cyclicity in the graph doesn't improve the optimality of the solution, but path short-cutting post procedure may help. The constructed trees are not too large as they are focused on linking two configurations, which ensures a rather light computational load. Many methods have been derived from the original RRT algorithm. They include Lower bound Tree RRT [124], Transition-based RRT [125], RRT* [122] (see Fig. 4.3.8), RRT# [126],

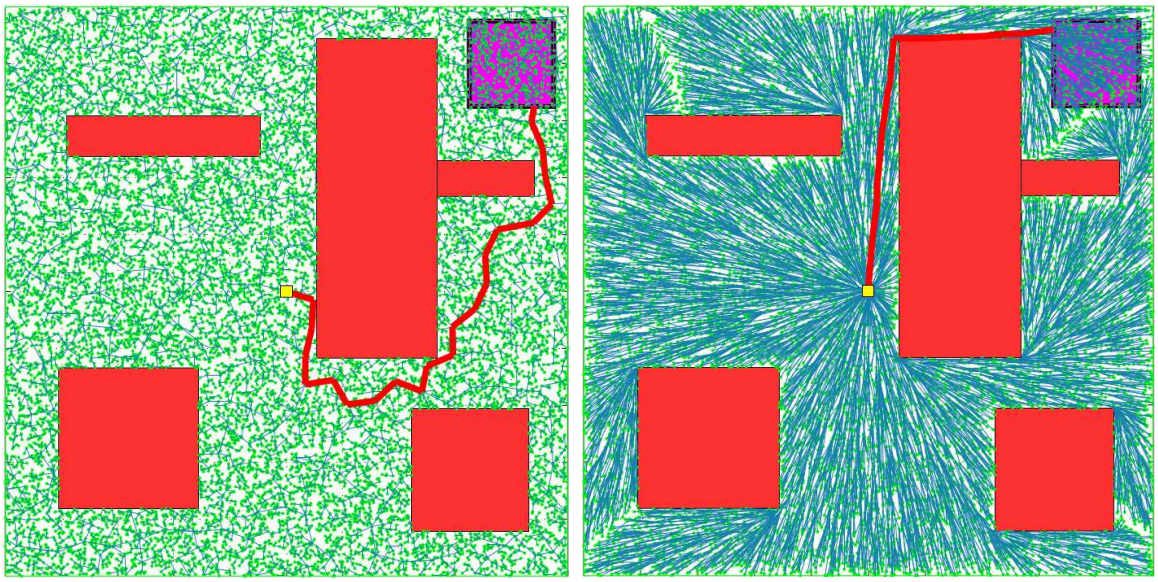


Figure 4.3.8: RRT and RRT*, for the same obstacle space and batch of sampled configurations. Credits to Sertac Karamen and Emilio Frazzoli, in [122], for the illustration.

informed RRT* [127], RRTX [128], and many others. The main purposes of these newer developments are the improvement of path optimality and/or computation time.

Probabilistic RoadMap (PRM) : The PRM algorithm is suited for multiple-query applications. This means that the PRM framework allows to solve several queries in quick succession. To be able to do that, the PRM framework builds and maintains a graph that aims at capturing the connectivity of the collision free space. This graph is referred to as the roadmap, or connectivity graph. The edges of this graph are weighted according to a user defined metric, which generally relates to a measure of the distance between the linked configurations.

After building this graph, queries are ready to be planned for. To do that, the start and goal configurations of the query are inserted to the graph by collision-checking trajectories between them, and neighbouring nodes of the graph. A graph search is then performed to find the ordered sequence of connected nodes which minimises

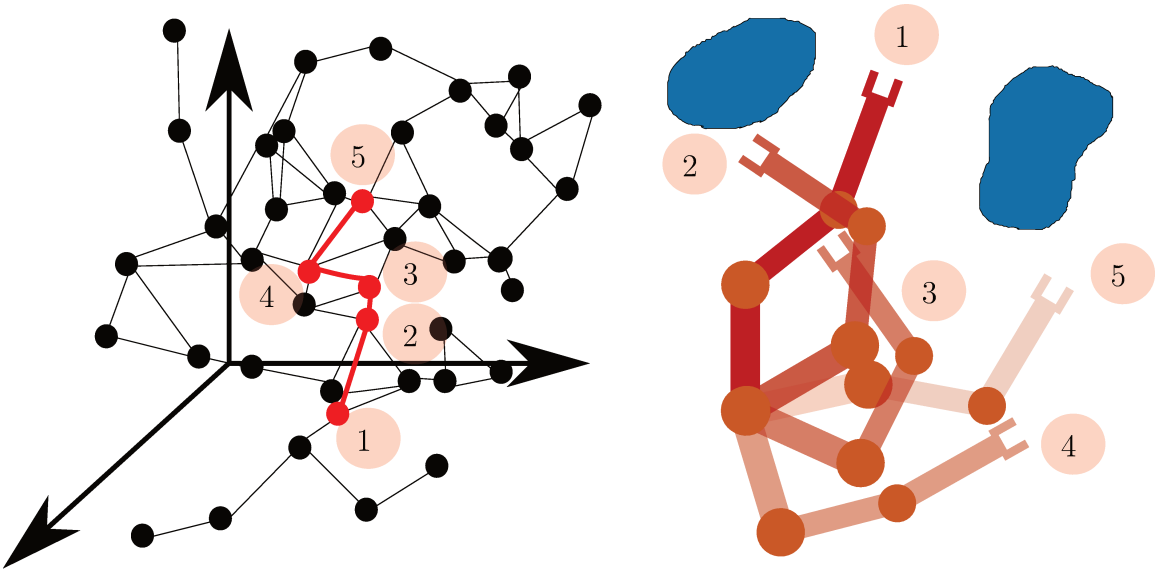


Figure 4.3.9: Application of a PRM algorithm to a serial system. The graph resulting from the offline phase of the PRM algorithm is used to plan a path between configuration #1 and configuration #5. The postures corresponding to the configurations shown in red on the left are the ones shown on the right of the figure.

the distance between the start and goal configuration. Famous shortest path graph search algorithms include the Breadth First Search [129–131], Dijkstra's [132–134] or A* algorithms [135].

Sampling-based methods have driven a lot of the motion planning researchers' attention in the past twenty years for some excellent properties which make them good candidates for industrial implementation. These methods are based on simple and intuitive concepts, which is an important leverage in the context of industrialisation. They are not computationally demanding, and thus provide results quickly. Their underlying principles are generalisable to any type of system, be it a mobile robot, an articulated robot, or a system composed of the two. Sampling-based methods are global approaches. Pure local approaches (like APF in Section 4.3.3) are generally prone to falling into local minima, while global ones take into account obstacles that may lay far ahead. Additionally, they are generally probabilistically complete and refinement of these methods are asymptotically optimal [122]. Probabilistic com-

pleteness ensures that if a collision free path exists, the probability that it is found by one of these methods approaches one exponentially with the number of nodes in the graph. Asymptotic optimality ensures that an optimal path will be approached given an increasing number of nodes in the graph. These properties are important because computational power increase will make them accordingly efficient.

4.4 Core concepts of PRM-based methods for dynamic environments

The goal of this section is to introduce some core concepts of PRM-based planners, which serve as foundation of the Receding Horizon Dynamic RoadMap (RH-DRM) planner that is described at the end of the chapter.

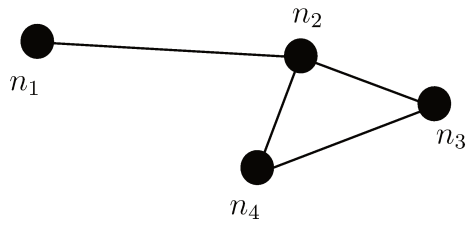
First, graph search methods will be introduced (Section 4.4.1). They are a core tool used by every PRM-based planners to find a shortest path within the connectivity graph. Then, some interesting approaches, based on the PRM planner and aiming at planning in dynamic environments, will be presented (Section 4.4.2). The algorithms associated to most of these tools and methods will be detailed to better understand the contributions of the RH-DRM.

4.4.1 Graph search algorithms

PRM methods rely on a representation of the configuration space connectivity, which is conveniently synthesised into a graph structure. Graphs are a very important tool of discrete mathematics. They conceptualise the notion of connections between pairs of objects. These objects are indifferently called vertices, nodes or points, while the connections between them are indifferently called edges, arcs, or lines. In the PRM framework, the nodes are obstacle-free configurations. The presence of an edge between two nodes in the roadmap represents the fact that the linear path between the two configurations associated to these nodes is collision-free. Representing the configurational connectivity as a graph allows to use some well-known algorithms that have been developed in the context of other graph theory problems. Graph theory

is at the crossroad of many fields, including communication networks, data organisation, linguistics, neuroscience, molecular representation, sociology, epidemiology, GPS-aided navigation, and many others.

A graph $\mathcal{G}$ is defined as a set of nodes and edges : $\mathcal{G}(\mathcal{N}, \mathcal{E})$. The nodes set $\mathcal{N}$ comprises N nodes, which are numbered from 1 to N. Nodes are referred to by their number, but each of them is related to a single object. In our case, these objects are obstacle-free configurations. The edges set $\mathcal{E}$ is composed of tuples that represent the connections between pairs of nodes. Each edge comprises a pair of nodes references (i, j), with $i, j \in \mathcal{N}$. Some graphs also weight each edge with a real number w_{ij}. In that case, edges are 3-tuples (i, j, w_{ij}). Edges can be directed, meaning that their connection is only effective in one direction, or undirected if a connection works either ways. $\mathcal{E}$ can often be represented as a matrix called the *connectivity matrix* $\boldsymbol{A}$ (sometimes called the *adjacency matrix*), as can be seen on Fig. 4.4.1. If the graph is unweighted, element $\boldsymbol{A}_{i,j}$ of $\boldsymbol{A}$ is either true (1) if a connection between node i and node j is effective, or false (0) if it isn't. If the graph is weighted, $\boldsymbol{A}_{i,j}$ may contain the weight of the edge connecting i to j, or 0 if no connection exists. If the graph is undirected, matrix $\boldsymbol{A}$ is symmetric because $\forall i \in \mathcal{N}, j \in \mathcal{N}, \boldsymbol{A}_{i,j} = \boldsymbol{A}_{j,i}$. The graphs representing the PRM we will be talking about in this thesis are undirected weighted graphs.



$$\boldsymbol{A} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix}$$

Figure 4.4.1: Example of a simple graph and its associated connectivity matrix.

Graphs and their analysis have raised a broad variety of generic mathematical problems. Graph colouring, visibility, covering, decomposition, traversal and route

problems are among these generic problems, whose solutions have been applied to different fields with success. Our own problem classifies into *route problems*, and is called the *shortest path problem*. The shortest path problem is the problem of finding the sequence of connected nodes, starting and ending in two given nodes (the query), that minimises the path cost, which is the sum of all the edges weight that are traversed.

Breadth First Search method : Among the most famous methods used to solve the shortest path problem, is the Breadth First Search (BFS) algorithm [129–131]. This method can be used in unweighted graphs, to find the shortest path in terms of the number of edges that are traversed. This method is however unable to take into account the fact that one edge might be more costly than another.

Dijkstra method (uniform search) : A method that generalises BFS is Dijkstra's algorithm [132–134]. This algorithm resembles BFS in many points, but uses the traversal cost of the edges instead of their number to expand outward. Dijkstra becomes equivalent to BFS if all the edges weight are unitary. These methods are relatively slow because they explore graphs uniformly from the seed node. To do so, each explored node maintains a variable that represents the minimal cost ever found for a path starting from the seed node and ending there. Additionally, each node keeps the reference of its best parent node, *i.e.* the one that is one step back in the same minimal cost path. The algorithm stops when the goal node is expanded. This strategy means that when the shortest path to the actual destination is found, all the shortest paths seeding from the starting node, and having a roughly equivalent cost, no matter where they end, have also been computed. A lot of computational work is done to explore solutions that are never going to be used.

A* method : A generalisation of the Dijkstra's algorithm is the A* algorithm [135]. This method uses a very similar approach to Dijkstra's, but stirs the exploration in nodes that seem more likely to be part of the shortest path leading to the destination. This method is guaranteed, like Dijkstra's, to find the shortest path in a weighted graph. It is also guaranteed to do so at least as fast as Dijkstra's, and generally

operates much faster in large graphs. The fundamental difference, in the A*, is the addition of the heuristic variable, which is associated to each node of the graph. This variable accounts for the optimistic estimated cost between each node and the goal node, no matter if they are connected by a single edge of not. Thanks to this variable, the exploration of the graph is biased towards the nodes that show the best potential of being among the shortest path sequence. Setting heuristic variables to zero for each node comes down to performing a Dijkstra's algorithm. The pseudo-code associated to the A* algorithm is presented in Algo. 1. In this version of the algorithm, each node s owns three attributes :

$s.g$: The cost of the shortest path that was found so far, seeding at the initial node and ending in s.

$s.bp$: The reference to the node that is the best parent of s (which is one step back in the best path ending in s, and found so far).

$s.h$: The value of the heuristic variable for s, which is the optimistic estimation of the cost from s to the goal node.

input :

- $\mathcal{G}(\mathcal{N}, \mathcal{E})$ // a graph ($\mathcal{G}$) comprised of nodes ($\mathcal{N}$) and edges ($\mathcal{E}$)
- sStart // the reference to the starting node
- sEnd // the reference to the destination node

output: path // an ordered list of node references

```
1  Procedure AStar(G(N,E), sStart, sEnd)
2      forall s ∈ N do
3          s.g ← ∞;
4          s.bp ← ∅;
5          s.h ← heuristicDistanceBetween(s, sEnd);
6      end
7      sStart.g ← 0;
8      open ← ∅, closed ← ∅;
9      add(sStart,open);
10     loop
11         s ← pop(open);
12         if s = sEnd then break loop ;
13         else if s = ∅ then return ∅ ;
14         foreach n ∈ neighbours(s,E), n ∉ closed do
15             costToNThroughS ← s.g + cost(s,n,E);
16             if costToNThroughS < n.g then
17                 n.g ← costToNThroughS;
18                 n.bp ← s ;
19                 if n ∉ open then add(n,open) ;
20             end
21         end
22         remove(s,open),add(s,closed);
23     end
24     s ← sEnd;
25     rpath ← ∅, append(s, rpath);
26     while s ≠ sStart do
27         p ← s.bp;
28         s ← p;
29         append(s, rpath);
30     end
31     path ← reverse(rpath);
32     return path;
33 end
34 Procedure pop(open)
35     s ← arg min_{c∈open}(c.g + c.h) ;
36     return s;
37 end
```

Algorithm 1: A* algorithm.

Initialisation : The A* algorithm begins by initialising each node attributes contained in graph $\mathcal{G}$ (lines 3 to 7). The g value of each node is set to infinity, except for the initial node, for which $g(\mathsf{sStart}) = 0$. This is explained by the fact

that the shortest path going from sStart to sStart has no cost at all. Function `heuristicDistanceBetween` (= hDB in Eq. (4.1) and Eq. (4.2)) computes an optimistic estimation of the distance between a node s and the destination node sEnd. This heuristic variable must be positive, and must abide to the triangular inequality :

$$\forall (s, s', s'') \in \mathcal{N}^3, hDB(s, s'') \leq hDB(s, s') + hDB(s', s''). \tag{4.1}$$

Additionally, to be *admissible*, this heuristic variable between two connected nodes must be bellow or equal to the cost of the edge connecting them:

$$\forall (s, s') \in \mathcal{N}^2 \text{ that are connected by an edge, } hDB(s, s') \leq cost(s, s', \mathcal{E}) \ . \tag{4.2}$$

Then, the algorithm initialises the promising nodes list *open* and stores the initial node sStart within (line 9).

Exploration : Then, the algorithm enters an infinite loop where it explores the graph. Within this loop, the algorithm *selects* a promising node and *expands* it. These two steps are described thereafter.

Selection : Within the loop, the algorithm first selects the most promising node from *open* (pop call, line 11). pop function finds the node s contained in *open* for which the sum $s.g + s.h$ is smallest (line 35). An important property of this algorithm is that when a node is selected, its g value is guaranteed to be as small as it can be, *i.e.* no path exists from sStart to s that is shorter than the one that could be reconstructed from the current heredity of s.

This sum adds up the cost of the shortest path going from sStart to s : $s.g$, and the optimistic estimation of the cost necessary to link sEnd from s : $s.h$. Therefore, this sum corresponds to an optimistic estimation of the cost of a path going from sStart to sEnd through s.

Expansion : Then, the expansion of the promising node s is done. Each of the neighbours of s, that has not yet been expanded (*i.e.* that is not in *closed*) is selected in turn. Then the cost of the path going from sStart to n through s is computed (line

15). If this path is the shortest that was ever found that linked sStart to n, then the value $n.g$ is updated to the correct one (line 17), and the best parent of n becomes s (line 18). Eventually, n is added to open (line 19). The term "expansion" is used to name this fundamental step of the A* algorithm because all the nodes that are connected to s (and that have never been in *open*) are added as candidates to *open*, which "expands" the search outwards from the seed.

Stopping conditions : The loop ends whenever pop returns nothing (line 13) or when it selects sEnd as the most promising node (line 12). In the former case, this means the *open* list is empty, and no other candidate exists to expand the search. Therefore, no solution exists that links sStart to sEnd. In the latter case, the selection of sEnd implies that the shortest path going from sStart to sEnd can be reconstructed by performing a heredity search.

Path reconstruction : When the break condition is triggered because sEnd was selected by the pop function, the path is ready to be reconstructed. The heredity of best parents, starting from sEnd, corresponds to the shortest path going from sEnd to sStart. Therefore, reversing its order returns the shortest path from sStart to sEnd. This reconstruction step begins at line 24 and ends at line 31.

4.4.2 Some PRM-based planners for dynamic environments

Despite their undeniable advantages, the sampling-based motion planner that were described in Section 4.3.6, were not designed specifically for dynamic environments.

The first part of this section will detail the PRM algorithm. Some intermediary conclusions will be driven to show the unsuitability of the original PRM versions for planning in dynamic environments. Then, the Dynamic RoadMap (DRM) algorithm will be described. The need for anticipation will then be stressed for, and an adaptation of PRM using anticipated obstacle trajectories will be described.

inputs :

- scGeom `// The static scene geometry`
- roGeom `// The robot geometry`
- nNodes `// The number of nodes`
- kNeighs `// The number of connections sought from each node`

output: A "k-neighbours simplified Probabilistic RoadMap (K-sPRM)" consisting in :

- $\mathcal{G}(\mathcal{N}, \mathcal{E})$ `// a graph` $\mathcal{G}$ `comprised of a node set` $\mathcal{N}$ `and edge set` $\mathcal{E}$

```
1  Procedure constructKsPRM(scGeom, roGeom, nNodes, kNeighs)
2      N ← ∅ ;
3      E ← ∅ ;
4      configs ← generateFreeConfigs(nNodes, scGeom, roGeom) ;
5      N ← createNodes(configs);
6      foreach c ∈ N do
7          cNeighs ← getKNearestNeighbours(c, N \ {c},kNeighs);
8          foreach n ∈ cNeighs, that has not yet been connected to c do
9              edgeCN ← createEdgeConnecting(c,n);
10             if isEdgeFree(edgeCN, scGeom, roGeom) then
11                 E ← E ∪ {edgeCN} ;
12             end
13         end
14     end
15     G ← G(N, E) ;
16     return G ;
17 end
```

Algorithm 2: K-sPRM construction algorithm.

4.4.2.1 The PRM algorithm

The simplified version sPRM [136] of the PRM algorithm (which is itself described in [121]) first consists of a pre-processing phase during which a connectivity graph is built. This pre-processing phase begins with an empty connectivity graph $\mathcal{G}(\mathcal{N} = \varnothing, \mathcal{E} = \varnothing)$. A batch of configurations $\mathbf{q}_{samp} \in \boldsymbol{\mathcal{C}}$ is sampled. The configurations that lie within $\boldsymbol{\mathcal{C}_{free}}$ are added to the nodes set[2] $\mathcal{N}$. Then, for each node of $\mathcal{N}$, connections to other nodes of $\mathcal{N}$ are attempted in order of increasing distance, until as far as a user-defined ball radius. An alternative version of the algorithm suggests to attempt connections to the $k \in \mathbb{N}^{+*}$ most neighbouring configurations[3]. An attempt of connection involves simulating the system travelling from one configuration to another. The system is simulated in all the configurations of a linear interpolation of the path and checked for obstacle-freedom. This connection and collision-freedom checking step is ensured by the *local planner* of the algorithm. If the local planner validates a path, the path reference is added to the edge set $\mathcal{E}$. In the original version of the algorithm [121], connections between nodes from the same connected component are avoided, to avoid cyclicity and quicken the pre-processing phase. The procedure used to construct the connectivity graph of a k-sPRM is detailled in Algo. 2 and illustrated in Fig. 4.4.2.

The most onerous pieces of the pre-processing phase are the collision checks that are performed by the local planner before edge insertion. The goal of this step is to check that no collision occurs while travelling from one configuration to another. It would be convenient to have access to the volume swept by the system during its traversal. It is however very difficult and costly to compute it in a continuous manner in the general case. Therefore, the typical approach is to interpolate the path and test the collisions in the interpolated configurations until a collision is detected or all configurations have been checked. The interpolation is made so that any

[2] To check that a configuration is within $\boldsymbol{\mathcal{C}_{free}}$, the system is simulated within its environment, in the posture corresponding to the configuration. If no collision or auto-collision is detected in the configuration, the test is successful and the configuration is added to the node set $\mathcal{N}$.

[3] This version of the algorithm, called the k-sPRM, is the one presented in this thesis.

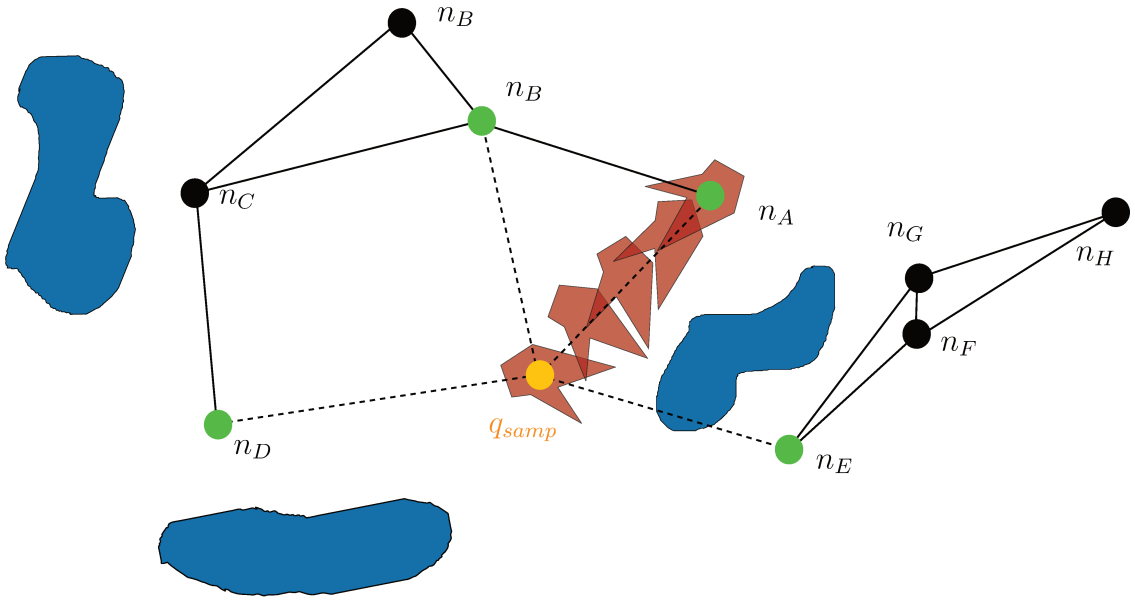


Figure 4.4.2: Illustration of Algo. 2. k-sPRM graph construction procedure for $k = 4$ (local planner collision testing). The 4 most neighbouring nodes (green nodes) of q_{samp} (yellow node) are identified (`getKNearestNeighbours` at line 7) and the local planner tests the paths leading to them (`isEdgeFree` at line 10). The blue areas correspond to the static obstacles envelopes, while the red shape is the agent's. In the illustration, connection from q_{samp} to n_A is validated by the local planner by testing several configuration interpolating the corresponding edge. On the other hand, a connection to n_E from q_{samp} will most probably fail. (*Note : In this illustration, the workspace is represented, and not the configuration space. The graph nodes correspond to the* (x, y, θ) *coordinates of the system but are merely displayed in* (x, y).)

Cartesian point attached to a rigid component of the system, which is posted at an interpolated configuration, remains "close" to its two counterparts of the next and previous interpolated configurations. "Close" generally means below a Cartesian distance d_{samp}. In practice, this can be ensured by taking sufficiently small configuration displacements between two interpolated samples. It is recommended, to unburden the pre-processing phase, to adopt a binary approach for the local planner checking approach [137]. A *binary approach* for collision checks of a path translate into halving the path and checking for collisions at the middle position. Until a collision is found, the algorithm recursively halves the pieces of the path and checks for collision.

This lasts until the length of the pieces is sufficiently small to validate the path or a collision is found. The reason why this method works better than the incremental approach is that middle positions are the ones that have the highest probabilistic chance of not being collision-free [138], being as far as possible and on the way of two collision-free configurations.

The context of dynamic environments brings new challenges. The connectivity graph that is constructed in the pre-processing phase is just suited to a single workspace topology. PRM are suited for planning multiple queries in the same environment. The pre-processing phase of the PRM algorithm is so costly that, unless the changes occur at a very slow pace, reconstructing the connectivity graph every time the environment changes is not an option. This would actually be utterly against the PRM philosophy, which consists of efficiently finding paths using a minimum-cost path search method on an established graph. Such a method would in fact be an unnecessarily expensive imitation of a RRT or RRG planner. A suitable, and philosophically consistent adaptation of the PRM method, for dynamic environments, should provide a way for the system to avoid dynamic obstacles without requiring the repeated use of the onerous local planner.

PRM-based methods do not require a representation of the obstacle configuration space $\mathcal{C}_{\mathbf{obs}}$. This is what makes them faster and much easier to generalise to high-dimensional problems than exact roadmaps methods (see Section 4.3.2). No explicit representation of $\mathcal{C}_{\mathbf{obs}}$ is done or maintained within the PRM framework. A major disadvantage of the original PRM approaches with regards to dynamic environments is that they base their research on the sole connectivity of the configuration free space $\mathcal{C}_{\mathbf{free}}$, while the dynamic obstacles geometry is naturally described within the workspace $\mathcal{W}$. In the absence of a generic and light mapping of $\mathcal{O} \subset \mathcal{W}$ into forbidden configuration space $\mathcal{C}_{\mathbf{obs}}$, the consequences of an obstacle displacement occurring in $\mathcal{W}$ on the topology of $\mathcal{C}_{\mathbf{free}}$ is not trivial at all.

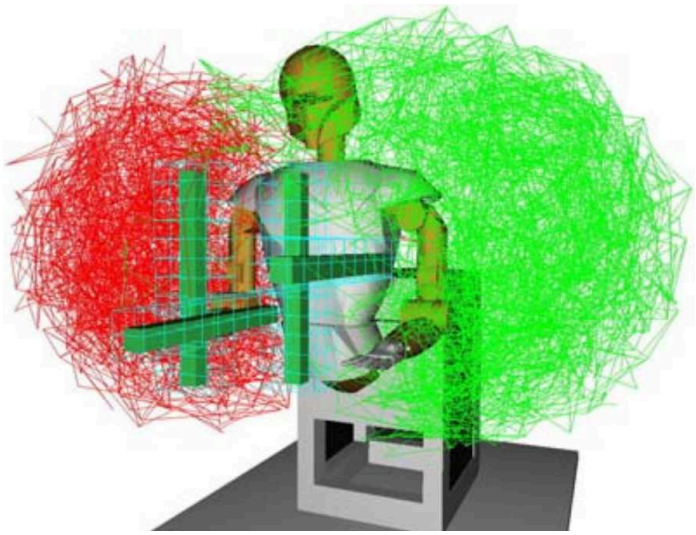


Figure 4.4.3: Dynamic roadmaps for a dual arm robot. Credit to Kallman and Mataric' [139] for the illustration.

4.4.2.2 The dynamic roadmaps approach

This section presents an adaptation of the PRM algorithm, called *Dynamic RoadMaps* (DRM) [139, 140]. This method was also successfully implemented in [141] with a 7-DOFs serial manipulator. The approach is based on the creation of a roadmap and on the creation of a discrete mapping of $\mathcal{O} \subset \mathcal{W}$ onto $\mathcal{C}$. This mapping is materialised by an *occupancy grid* covering the entire workspace. This grid is composed of voxels when $\mathcal{W}$ is 3-dimensional and pixels when $\mathcal{W}$ is 2-dimensional[4]. This mapping is created in parallel of the roadmap construction. It consists of associating the reference of each sampled configuration (and of each collision-free edge) to each voxel that is -at least- partially occupied by the system when simulated in (or along) them. A discrete description of the system workspace occupancy is, in this manner, built with regards to its configuration space position.

This mapping is then used during the online phase to discard nodes and edges of the connectivity graph. First, the dynamic obstacles voxels occupancy is determined.

[4]For the sake of brevity, we will talk about voxels, but the same principles apply to 2D workspaces, with pixels.

Then, all the nodes and edges which are associated to these occupied voxels are identified. They are then set aside and ignored during the graph search procedure that is performed on the connectivity graph to find the shortest path.

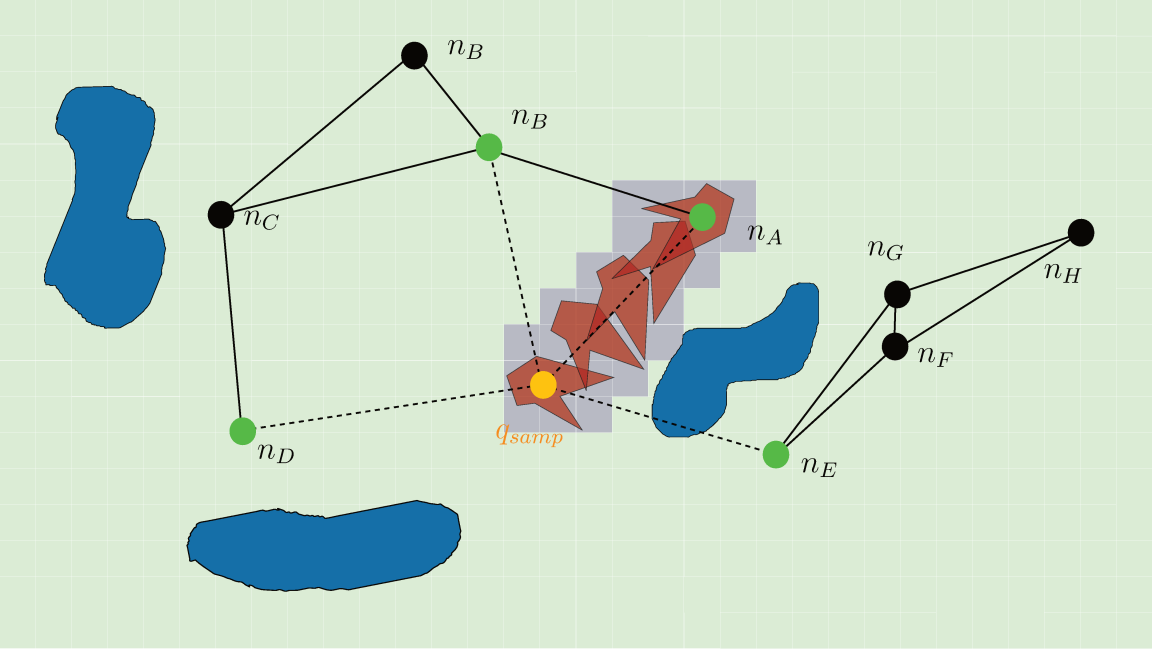


Figure 4.4.4: Illustration of Algo. 3. DRM pre-processing phase (local planner testing and discrete mapping to occupancy grid). Just as in the PRM procedure, edges freedom is tested thanks to the local planner. Additionally, the occupancy of the edge is determined along the way (`sweepEdge` function at line 14) with regard to the occupancy grid. Edge-swept pixels are displayed in light purple while unconcerned ones are in light green.

DRM thus provide an efficient way of reacting to dynamic changes in the environment. Compared to PRM, the pre-processing phase is more expensive. It requires testing collisions with each voxels for every checked configurations and edges. Fortunately, the pre-processing phase is an offline phase, and thus do not affect the efficiency of the planner when it is actually used. The online phase is very little impacted by this new framework, once the occupancy of the workspace is determined. The discarding of the edges and nodes concerned by the occupancy of space is

inputs :

- scGeom // The static scene geometry
- roGeom // The robot geometry
- occGrid // An occupancy grid
- nNodes // The number of nodes
- kNeighs // The number of connexions sought from each node

output: A "Dynamic RoadMap" consisting in :

- $\mathcal{G}(\mathcal{N},\mathcal{E})$ // a graph ($\mathcal{G}$) comprised of nodes ($\mathcal{N}$) and edges ($\mathcal{E}$)
- occNodes // mapping between $\mathcal{N}$ and occGrid
- occEdges // mapping between $\mathcal{E}$ and occGrid

```
1  Procedure contructDRM(scGeom, roGeom, occGrid, nNodes, kNeighs)
2      N ← ∅ ;
3      E ← ∅ ;
4      occNodes ← ∅ ;
5      occEdges ← ∅ ;
6      configs ← generateFreeConfigs(nNodes, scGeom, roGeom) ;
7      N ← createNodes(configs);
8      foreach c ∈ N do
9          occupiedPixels ← pixelsOccupiedBy(c, roGeom, occGrid) ;
10         linkNodeAndPixels(occNodes,c, occupiedPixels) ;
11         cNeighs ← getKNearestNeighbours(c,N \ {c},kNeighs);
12         foreach n ∈ cNeighs, that has not yet been connected to c do
13             edgeCN ← createEdgeConnecting(c,n);
14             [isEdgeFree, sweptPixels] ← sweepEdge(edgeCN, scGeom, roGeom,
                occGrid);
15             if isEdgeFree then
16                 E ← E ∪ {edgeCN} ;
17                 linkEdgeAndPixels(occEdges,edgeCN, sweptPixels) ;
18             end
19         end
20     end
21     G ← G(N,E) ;
22     return [G, occNodes, occEdges] ;
23 end
```

Algorithm 3: Dynamic roadmap (DRM) construction. Changes with regards to the PRM construction (Algo. 2) are highlighted in deep blue.

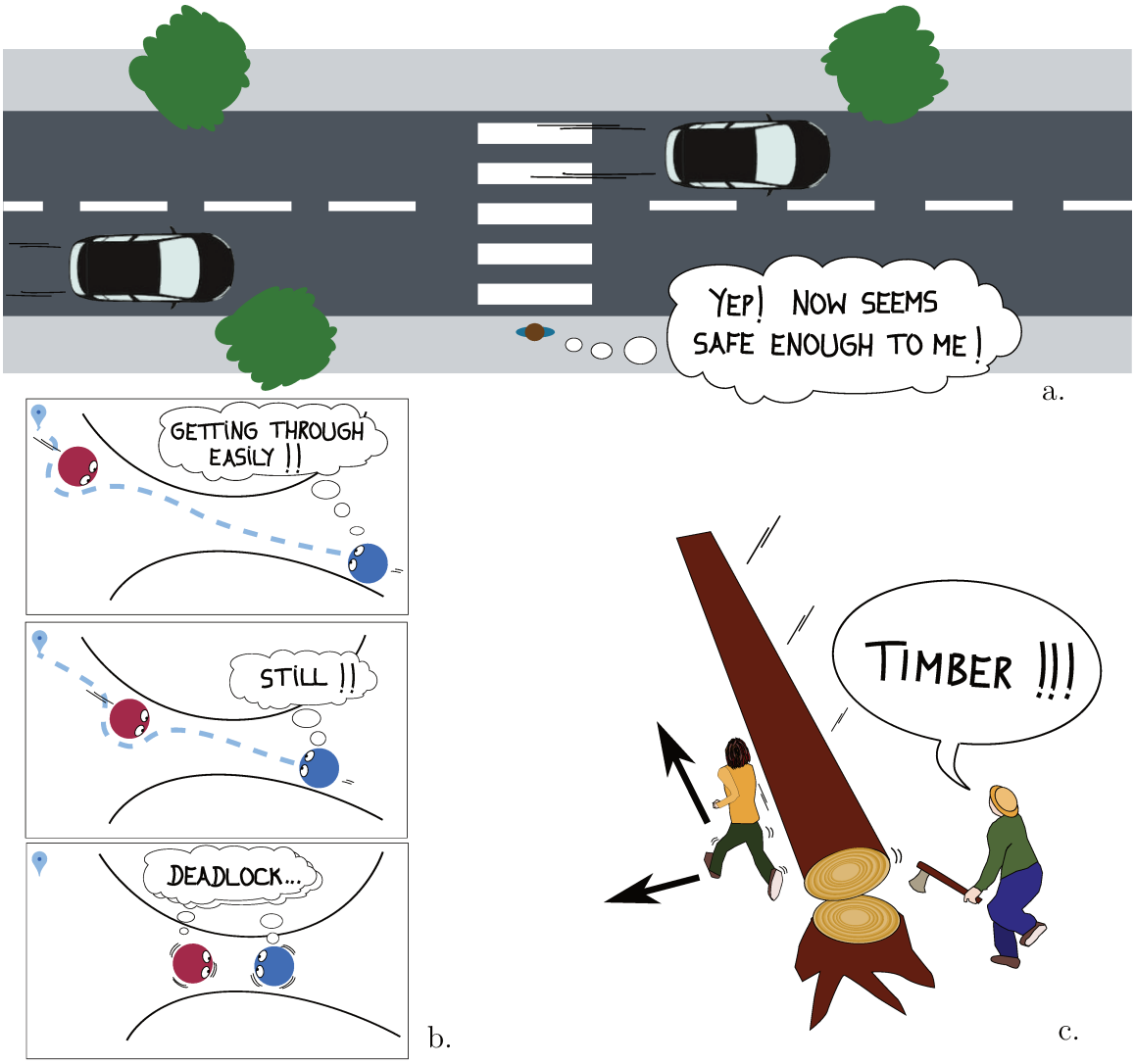


Figure 4.4.5: Illustrative examples where environmental motions anticipation is paramount. Not anticipating environmental motions may lead to deadlocks (b.), or even dangerous behaviours (a. and c.).

very light. The strategy of the DRM is therefore a very interesting one. It allows to take into account the dynamic obstacles in the planning phase without adding much computational load. It retains all the advantages of the PRM approach, and allows more.

A major limitation of the DRM framework, however, is that the planned path only takes into account the occupation of the obstacles in a single instant, and completely ignores their undergoing motion. This comes down to having paths computed as if the dynamic obstacles were remaining in their current position. The future configurations of the system are therefore planned without considering the future space occupancy of the obstacles.

Just like many other sampling-based methods, the DRM approach is global with regards to space. It has the capacity to plan paths globally (in a spatial manner of speaking), instead of only reacting to what lies in the direct vicinity. However, this framework plans locally in a temporal way of seeing things, as it only looks at the current - *i.e.* local with regards to time - occupancy.

This local-time assumption is somewhat correct if the agent's velocity is very high compared to the velocity of the obstacles. However, in our case, the dynamic obstacles are human beings or counterpart systems. Therefore, the velocities involved are relatively close to one another, and the obstacles can't be considered as if they were remaining in the same place.

Fig. 4.4.5 illustrates situations were anticipation is needed but not used to plan motions. These examples may suggest what using dynamic roadmaps in an industrial environment could lead to. Deadlock occurrences are a recurring problem that could easily arise if motions were planned without predicting obstacle motions. Dangerous situations could also occur, leading to collisions, breakage and injuries.

Generally speaking, anticipation becomes compulsory in motion planning problems where the obstacles motions have a non-negligible velocity compared to the system's one. To a certain extent, motion prediction even allows to plan paths in environments

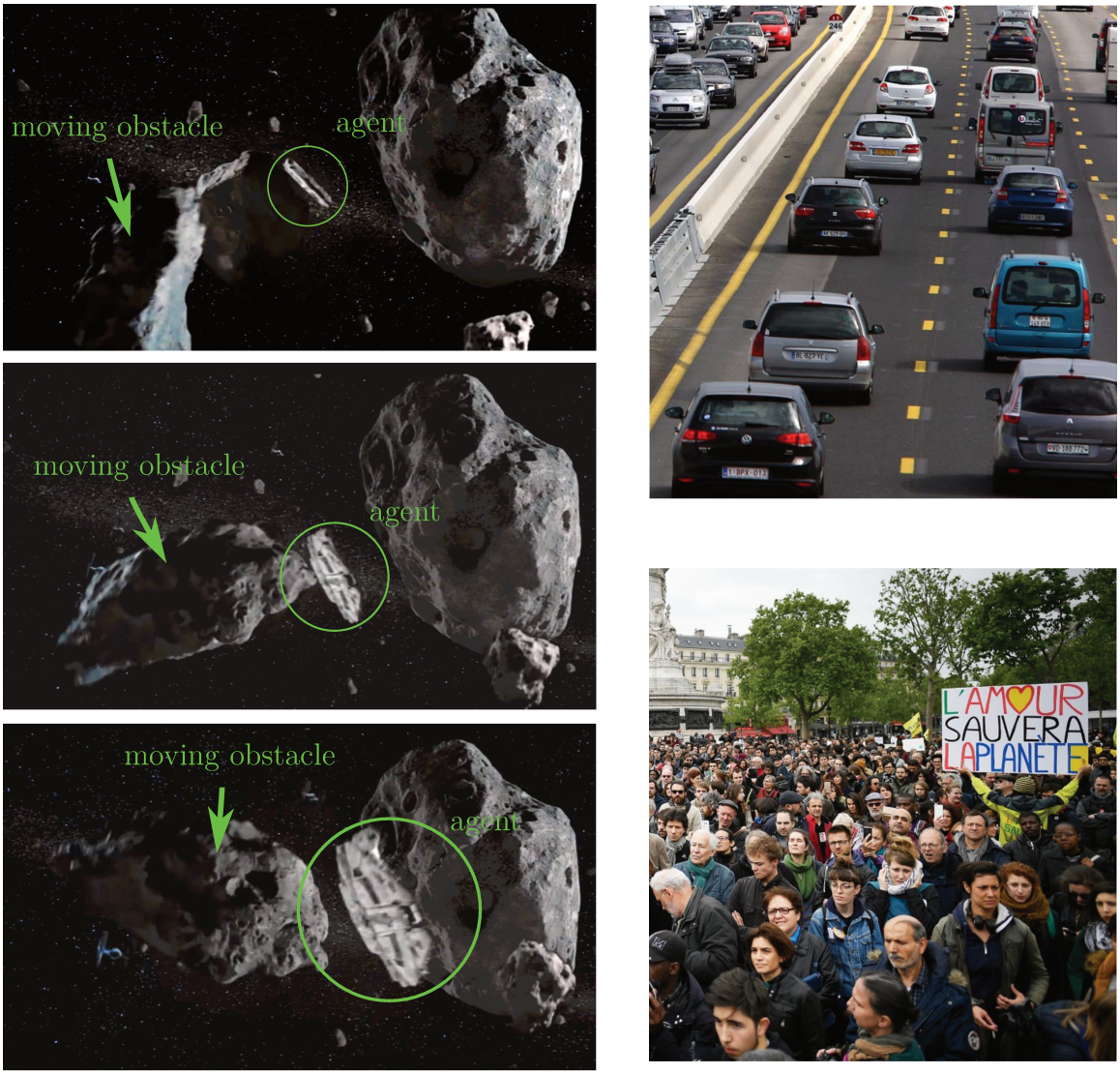


Figure 4.4.6: Real life examples of motion planning using anticipation. Han Solo and Chewbacca [142] would certainly not have made it through the asteroid field without anticipating obstacles motions. Moving within crowds or navigating on highways requires more than a snapshot of the space occupancy to be performed. Only the anticipation of the surrounding obstacles motions allows it.

where the obstacles have a higher velocity than the agent itself. Many examples show that humans use a notion of anticipation to plan their motions, like in Fig. 4.4.6.

4.4.2.3 A PRM-based method that uses anticipation

Another interesting approach, which is based on the PRM framework, was presented in [143]. The originality of this approach with regards to our problem lies in the notion of obstacle motions anticipation. The problem solved in this paper is the one of planning a collision-free path for a mobile robot, within a partially known environment where the trajectories of the dynamic obstacles are predicted or guessed.

The algorithm begins, just like a classic PRM, with the creation of a roadmap within the known static environment. The objective in this initial phase is to build a graph that is extensive enough to provide low-cost paths (with regard to time traversal). Graph cyclicity is allowed to provide with many alternative routes when obstacles obstruct portions of the graph.

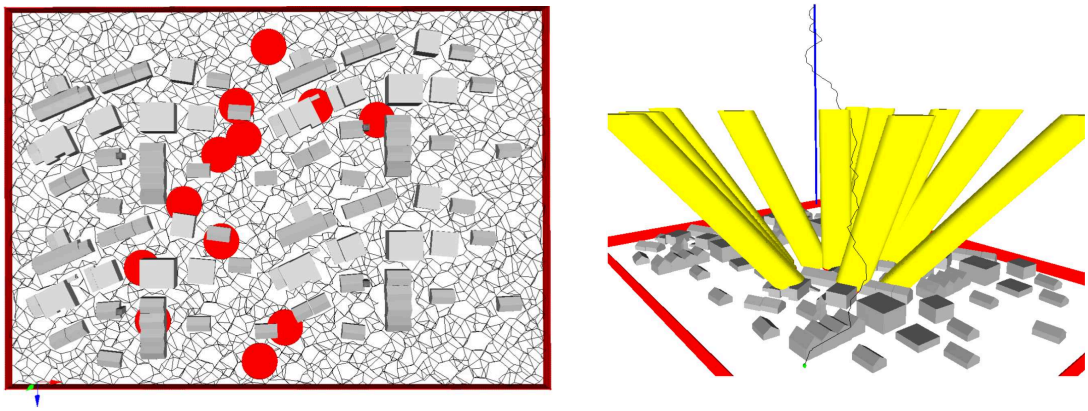


Figure 4.4.7: Roadmap and dynamic obstacles positions and predicted trajectories. Illustration credit to Van den Berg et al. in [143].

After that, the authors determine the states of the graph for a sampling of linearly distributed dates (separated by a duration δt), based on the prediction of the trajectories of the obstacles. All these graphs are stacked up along what can be seen as

the time axis, and each node is, obstacle-freedom allowing, linked to its predecessor and successor with regards to time. To establish the states of the graphs, all nodes and edges situated at a given distance of an obstacle at a given instant are discarded, which comes down to simplifying the shapes of the agent and obstacles to circles for the online planning. After building this graph, an anytime graph search called Anytime D* (AD*) [144] is performed to find a path to the destination. This graph search approach is used because the graph built is very big and performing an A* search on it would take too much time to be used online. Therefore, the idea here is to reuse the planned path of the previous search and adapt it to the new graph topology, which is said to be much faster. This approach allows the agent to use each node or edge several times, and to wait in a node if necessary. These behaviours are essential when time is involved.

In this approach, the stacking of graphs on a time axis is very interesting, because it allows anticipation to be part of the scheme. A drawback of this stacking is that no optimal solution can be found unless δt is infinitely small, which leads to an infinitely big graph, whose size is directly correlated to the computational load. Additionally, nodes and edges discarding uses a very basic strategy, which doesn't allow to take into account complex geometries for obstacles or agent, let alone be used for serial robots.

4.5 Receding horizon dynamic roadmaps (RH-DRM)

4.5.1 Introduction

The novel motion planning strategy described in this chapter is called the *Receding Horizon Dynamic RoadMaps* (RH-DRM) method. Anticipation is a key feature for a safe, nimble and efficient trajectory.

Using DRM to predict graph topology : RH-DRM extends the concept of dynamic roadmaps (DRM) to an anticipated occupancy of space over a receding time

horizon[5]. The core advantages of DRM are kept. The method is still generic to serial and mobile systems. It still creates and uses a discrete mapping between obstacle and configuration space, remains efficient and abides to simple and sensible rules.

The pre-processing phase of the RH-DRM is exactly the same as the one of the DRM, which is detailed in Section 4.4.2.2. The roadmap is constructed, nodes and edges are tested with a local planner, and associated to voxels of an occupancy grid.

Given an online prediction of the moving obstacles trajectories, the prediction of the voxels occupancy is determined. From this anticipated occupancy, the topology of the connectivity graph is predicted through the mapping between occupancy grid and graph that was created during the pre-processing phase.

Objective : An *optimal journey* is to be found within this *changing graph*, whose nodes and edges may be *unavailable for periods of time*, and for which *edges traversal is not instantaneous*.

Graph representation : The novel strategy described in this chapter is based on a type of graph that was named *step-in interval graph*. A step-in interval graph intuitively and efficiently represent changes in the topology of a static graph while accounting for non instantaneous edges traversal. Additionally, an adaptation of the A* algorithm is presented to solve the earliest arrival journey problem within this type of graph, for non instantaneous edge traversals. The introduction to these graphs is made in Section 4.5.2.

Optimal journeys : The involvement of time and changes within the graph completely changes the paradigm. Three solutions could be sought: the shortest journey, the fastest journey and the foremost journey [145]. The shortest journey solution is the journey within this evolving graph for which the travelled spatial length is minimised, regardless of its duration. The fastest journey instead minimises the time needed to travel from one node to another, regardless of the date where the agent

[5] As frequently as possible, the planning is performed according to the space occupancy prediction which is given over a fixed-time horizon.

begins to move and of the overall distance. Lastly, The foremost journey solution gives the journey that allows to arrive as early as possible at the destination.

Real life examples illustrating these problems are numerous. For instance, an elderly person might want to reach a destination across a moving crowd while minimising the number of steps taken, regardless of the time needed to journey, thus needing a shortest journey solution. A pedestrian might want to minimise the time of a journey if rain pours over, by choosing the optimal moment to start their journey, thus needing a fastest journey solution. Lastly, someone who is in a great hurry will look for the journey that enables them as soon as possible, thus needing the foremost journey.

For our problem, we will consider the foremost journey, because we want our system to arrive as early as possible to its destination and have no choice in the starting time, but to begin instantly. The foremost journey may admit stops, multiple runs in a single node, multiple traversals of a single edge, which cannot be computed using the standard version of A* described in Algo. 1. The computation of the foremost journey will be described in Section 4.5.3.

4.5.2 An adaptation of interval graphs

In our problem, a predicted partial occupation of a pixel/voxel by a moving obstacle will translate into the temporary unavailability of all nodes and edges that are associated to this pixel/voxel. To account for the changes in the topology of graphs, the field of *temporal networks* [146] was created. A type of temporal network called *interval graphs* can be used to represent a graph whose nodes and edges may be unavailable for periods of time. Interval graphs are used in operational research and scheduling theory for resource allocation problems. Such an interval graph is depicted in Fig. 4.5.1 a).

Another difficulty in our problem is the fact that edges traversals are not instantaneous. To ensure that the edges availability will be sustained throughout their entire traversal, the intervals associated to the availability of the edges can be shortened by the time needed to traverse them. This preserves the intervals of time where the

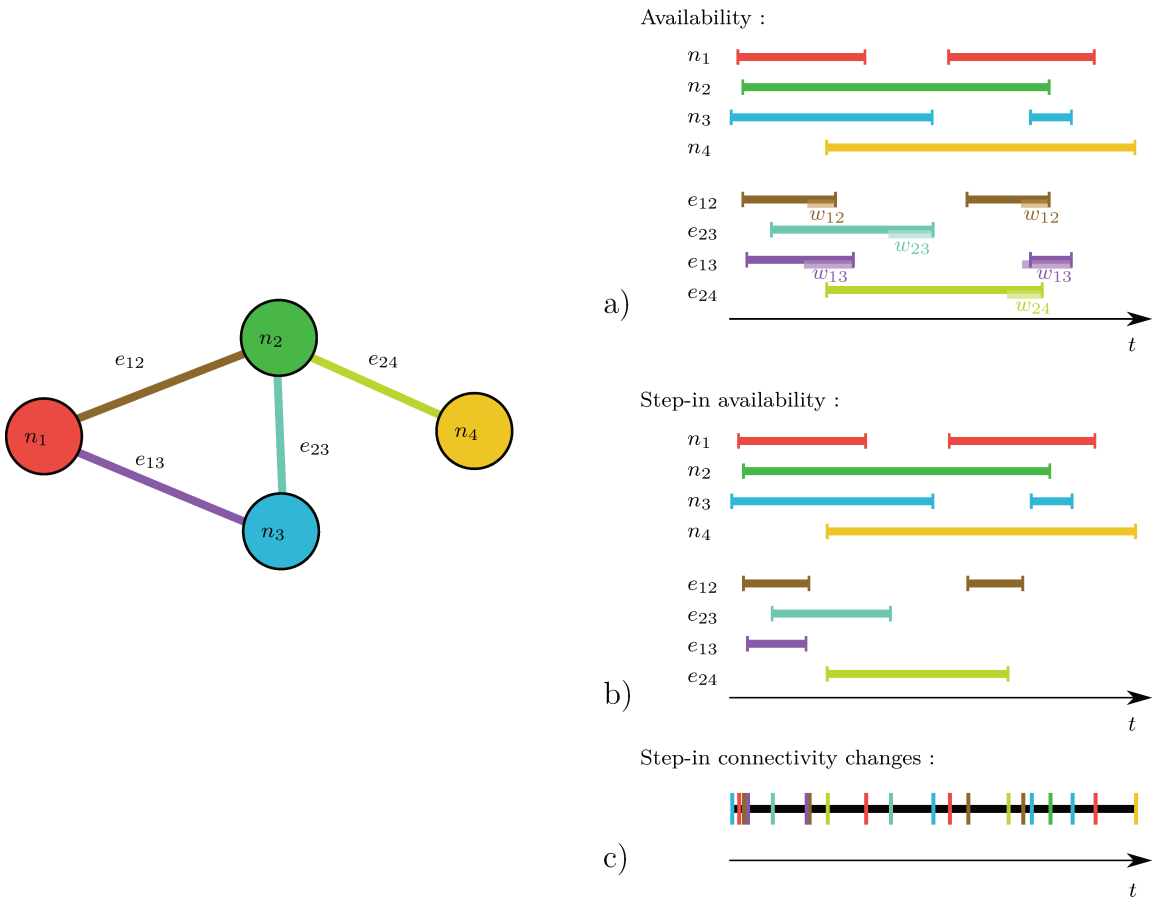


Figure 4.5.1: a) A graphical representation of an interval graph, that shows nodes $(n_i)_{i\in[\![1..4]\!]}$ and edges $(e_{ij})_{i,j\in[\![1..4]\!]^2}$ availability. The weights (which are also the traversal durations) of the edges are the corresponding $(w_{ij})_{i,j\in[\![1..4]\!]^2}$.
b) A graphical representation of the corresponding step-in interval graph. The availability of the edges is shortened by their corresponding traversal duration $(w_{ij})_{i,j\in[\![1..4]\!]^2}$.
c) The graph step-in connectivity changes occur at each vertical tick displayed here. The graph connectivity remains constant between two successive ticks.

system can begin to use the edges. We name such an interval graph a *step-in interval graph* (see Fig. 4.5.1 b)), because the intervals related to the edges represent the moments when the agent is allowed to begin edges traversal (*i.e.* when edges can be "stepped in"). In our framework, *the weight associated to an edge correspond to the estimated time required to traverse it.* If an availability interval is shorter than the traversal duration, then this interval must be discarded as an impracticable one for this edge (on Fig. 4.5.1 b), e_{13} second period of availability is discarded).

The ordered sequence of all intervals beginnings and ends $\mathcal{D} = \{t_0, \ldots, t_{N_0}\}$, that can be seen in Fig. 4.5.1 c), gives a very valuable piece of information. Each interval comprised between a pair of successive dates $[t_i, t_{i+1}[$, (with $i \in [\![0..N_0 - 1]\!]$), corresponds to a period where the graph *step-in connectivity* is constant. This means that during each of these intervals, the graph connectivity doesn't change. In each of the dates of $\mathcal{D}$, the graph's topology changes.

The overall step-in connectivity can be mathematically described by a set $\mathcal{G}(\mathcal{N}, \mathcal{E}, \mathcal{D})$. In $\mathcal{G}$, $\mathcal{N}$ is the set of nodes, which are each a logical (N_0)-tuple associated to a configuration. Each of the components of the (N_0)-tuple determines the availability of a node between each interval defined by a pair of successive dates of $\mathcal{D}$. $\mathcal{E}$ is the set of edges, which are each a $(N_0 + 3)$-tuple. N_0 elements correspond to the availability of the edge between each interval defined by a pair of successive dates of $\mathcal{D}$. One element corresponds to the weight of the edge, and two elements correspond to the node references that the edge connects.

This temporal network can be seen as the succession of N_0 static graphs, that remain constant within a time interval $[t_i, t_{i+1}[$ (with $i \in [\![0..N_0 - 1]\!]$). Each of these graphs can be represented by a step-in adjacency matrix $\boldsymbol{A}^{[t_i,t_{i+1}[}$. This matrix describes the step-in connectivity of each node during the interval $[t_i, t_{i+1}[$ to which the graph is associated. This matrix is sometimes referred to as the *graphlet* [147, 148], or *snapshot representation* [149]. We will call it the *interval-graphlet* to be more specific. In a network comprised of N nodes, the non diagonal elements $a_{k,l}^{[t_i,t_{i+1}[} : k, l \in [\![1..N]\!]^2 \ \& \ k \neq l$ of interval-graphlet $\boldsymbol{A}^{[t_i,t_{i+1}[}$, correspond to the step-in connectivity

of an edge connecting node k to node l. The diagonal terms $a_{k,k}^{[t_i,t_{i+1}[} : k \in [\![1..N]\!]$ determine the availability of a node within the time interval $[t_i, t_{i+1}[$.

4.5.3 Computing the foremost journey

The foremost journey within these types of interval graphs can be computed using Algo. 4. This algorithm uses a similar strategy to A*'s, and adapts it to the step-in interval type graphs, which was presented in the previous section. Changes from Algo. 1 are displayed in deep blue. A special effort was made to keep the version of the algorithm as similar as possible from the original A* algorithm.

The main difference is the type of nodes that is maintained. Instead of exploring nodes directly, the algorithm explores the temporal versions of these nodes. They will be referred to as *node temporal instances (NTI)*. For each interval of $\mathcal{D}$, one temporal instance of each node is created (line 5) and maintained. A NTI is only valid for its associated period of time $[t_i, t_{i+1}[$. Its step-in connectivity can in practice be determined thanks to the interval-graphlet $\boldsymbol{A}^{[t_i,t_{i+1}[}$ and is performed by function `stepInNeighbours` (line 14). The temporal version s_t of a node s can be reached thanks to the function `getTemporalInstance(`s`,`t`)`, where s is the node reference, and t the time for which $t \in [t_i, t_{i+1}[$, $[t_i, t_{i+1}[$ being the validity interval of s_t. The implication of time within the scheme operates some fundamental changes (lines 16-25), because a NTI can be left from the moment it is reached until its time interval ends. Therefore, the expansion of one NTI s_t is done towards each NTI of the step-in neighbours of s_t that can be reached until the last moment of validity of s_t.

input :

- $\mathcal{G}(\mathcal{N},\mathcal{E},\mathcal{D})$ // a step-in interval graph ($\mathcal{G}$) comprised of nodes ($\mathcal{N}$), of edges ($\mathcal{E}$) and dates set ($\mathcal{D}$).
- sStart // the reference to the starting node
- sEnd // the reference to the destination node

output: path // an ordered list of nodes associated to a timing

```
1  Procedure AStarIntervalGraph(G(N,E,D), sStart, sEnd)
2      forall s ∈ N do
3          heur ← heuristicDistanceBetween(s, sEnd);
4          forall [t_i, t_{i+1}[ ∈ D do
5              createTemporalInstance(s,[t_i, t_{i+1}[,heur,E);
6          end
7      end
8      open ← ∅, closed ← ∅;
9      s_t ← getTemporalInstance(sStart, 0), s_t.g ← 0, add(s_t,open);
10     loop
11         s_t ← pop(open);
12         if s_t.id = sEnd then break loop;
13         else if s = ∅ then return ∅;
14         foreach n ∈ stepInNeighboursOf(s_t,E) do
15             costEdge ← cost(s_t.id, n, E);
16             firstArrival ← s_t.g + costEdge, lastArrival ← s_t.u + costEdge ;
17             t ← firstArrival;
18             while t < lastArrival do
19                 n_t ← getTemporalInstance(n, t) ;
20                 if n_t ∉ closed, t < n_t.g then
21                     n_t.g ← t;
22                     n_t.bp ← s_t;
23                     if n_t ∉ open then add(n_t,open);
24                 end
25                 t ← n_t.u;
26             end
27         end
28         if canStayIn(s_t) then expandToNextTemporalInstance(s_t);
29         remove(s_t,open), add(s_t,closed);
30     end
31     rpath ← ∅, append(s_t, rpath);
32     while s_t.id ≠ sStart do
33         p_t ← s_t.bp;
34         s_t ← p_t;
35         append(s_t, rpath);
36     end
37     path ← reverse(rpath) , return path ;
38 end
```

Algorithm 4: Modified A* algorithm for step-in interval graphs with non-instantaneous edge traversal duration. Changes from Algo. 1 are displayed in deep blue.

```
1  Procedure createTemporalInstance(s,[t_i, t_{i+1}[,heur,E)
2      s_t ← ∅, s_t.id ← s, s_t.l ← t_i, s_t.u ← t_{i+1}, s_t.h ← heur, s_t.g ← ∞, s_t.bp ← ∅;
3      linkSandSt(s,s_t);
4  end
5  Procedure pop(open)
6      s_t ← arg min_{c_t ∈ open}(c_t.g + c_t.h);
7      return s_t;
8  end
9  Procedure expandToNextTemporalInstance(s_t)
10     if s_t.u ≠ ∞ then
11         c_t ← getTemporalInstance(s_t.id,c_t.l);
12         c_t.g ← c_t.l;
13         c_t.bp ← s_t;
14         if c_t ∉ open then add(c_t,open);
15     end
16 end
```

Algorithm 5: Implementation of `createTemporalInstance`, `pop` and `expandToNextTemporalInstance` procedures, which are used in the A* algorithm for step-in interval graphs.

A NTI s_t owns several attributes:

$s_t.id$: The node reference s which is associated to the node temporal instance s_t.

$s_t.l$: The lower boundary (t_i) of the interval which is associated to s_t.

$s_t.u$: The upper boundary (t_{i+1}) of the interval which is associated to s_t.

$s_t.g$: The cost of the foremost journey (*i.e.* the duration of the best journey) that was found so far, seeding at the initial node at time 0 and ending in s_t.

$s_t.bp$: The reference to the NTI that is the best parent of s_t (which is one step back in the best journey ending in s_t, and found so far).

$s_t.h$: The value of the heuristic variable for s_t, which is the optimistic estimated duration of the journey from $s_t.id$ to $sEnd$.

Initialisation : The algorithm starts by initialising the NTI of each node of $\mathcal{N}$ for each interval of $\mathcal{D}$. The g value of each instance is set to ∞, except for the temporal instance of $sStart$ at time $t = 0$, for which $g = 0$. The heuristic distance to the destination is the same for each temporal instance of a node, and is thus computed once at line 3. Then, the starting node temporal instance is added to the promising list *open*.

Exploration : Then, the algorithm enters an infinite loop where it explores the graph. Within this loop, the algorithm *selects* the most promising NTI, and *expands* it towards all its step-in neighbours. These two steps are described thereafter.

Selection : Within the loop, the algorithm first selects the most promising NTI from *open* (`pop` call at line 11). `pop` function finds the NTI s_t contained in *open* for which the sum $s_t.g + s_t.h$ is smallest (line 6). This sum adds up the duration of the shortest journey from the initial temporal instance of $sStart$ to the current NTI ($s_t.g$) with the optimistic estimation of the duration necessary to link $s_t.id$ to $sEnd$ ($s_t.h$). Therefore, this sum corresponds to an optimistic estimation of the duration

of a journey going from $sStart$ at time $t = 0$ to $sEnd$ through $s_t.id$.

Expansion : Then, each step-in neighbour of s_t, is identified at line 14. The date of the earliest arrival ($firstArrival$) and latest arrival ($lastArrival$) to this neighbour, taking a journey from $sStart$ at time $t = 0$ to there, through s_t, is computed (line 16). Then, all the temporal instances of this neighbour, which are valid between $firstArrival$ and $lastArrival$, are added to *open* if they weren't in it yet. Their g value and best parent bp are updated if passing through s_t to reach them provides an earliest arrival journey (line 21-22).

After expanding to all step-in neighbours, the temporal instance of $s_t.id$, corresponding to the interval following s_t's is being expanded to if the agent is allowed to stay there (line 28). If a NTI - which is already in *open* - is being expanded to from its preceding temporal instance, then the best parentage goes in priority to the preceding temporal instance. This prioritisation has an effect on where the agent chooses to stop in priority when two equivalently optimal journeys have the same geometric path description but different schedules. Doing so forces the agent to advance as far as possible in its geometric path before choosing to stop.

Stopping condition : The loop ends whenever `pop` returns nothing (line 12) or when it selects a temporal instance of $sEnd$ as the most promising NTI (line 13). In the former case, this means the *open* list is empty, and no other candidate exists to expand the search. Therefore, no solution exits that links $sStart$ to $sEnd$. In the latter case, the selection of an instance of $sEnd$ implies that the foremost journey going from $sStart$ to $sEnd$ can be reconstructed by performing a heredity search.

Path reconstruction : When the break condition of line 13 is triggered, the path is ready to be reconstructed (lines 31-37). The temporal instance of $sEnd$, which was stored within s_t at the last call of `pop` is added to the ordered set $rpath$. The heredity of best parents, starting from $sEnd$, eventually leads to the initial instance of $sStart$. Therefore, reversing the order of $rpath$ (line 37) returns the foremost path between $sStart$ and $sEnd$. The duration of the stops can easily be computed by comparing the g values of two successive nodes with the cost of the edge linking them.

4.5.4 An illustrative example

To illustrate the inner-workings of the algorithm, let us consider the motion planning problem displayed at the top of Fig. 4.5.2. In this problem, the agent has to travel from the left extremity of a corridor to the right extremity (*i.e.* from n_1 to n_4). However, a moving obstacle is preventing the agent from performing a trivial motion to the destination by blocking the way and advancing towards the former. The spatial occupation of the moving obstacle is predicted over a 10 seconds time horizon, and displayed using a time-colour scale (from deep blue at $t_0 = 0.0s$ to white at $t_5 = 10s$). Each node owns 2 attributes : its heuristic value (which is contained in h), and its availability interval(s)). Each edge owns 2 attributes : its weight (which is contained in w_{ij}), and its step-in availability[6]. The step-in availability of an edge is computed by shortening its availability intervals with its weight, as was explained in Section 4.5.2. The knowledge of the availability of each node of the step-in availability of each edge allows to draw (at the bottom of Fig. 4.5.2) the availability and step-in availability intervals for all nodes and edges, as in Fig. 4.5.1 b).

For each period $P_i = [t_i, t_{i+1}[$ where the step-in connectivity of the graph is constant, an interval-graphlet $\boldsymbol{A}^{[t_i,t_{i+1}[}$ can be computed. All the interval-graphlets of our problem are displayed on Fig. 4.5.3. On the same figure appears a modified-slice-graph representation of our step-in interval graph from $t = t_0$ to $t = t_6$.

On this slice graph representation, the NTI objects, that were extensively used to explain Algo. 4, are represented with the coloured dots ● ● ● ●. All temporal instances of a node lie at the same vertical level (horizontal alignment). Each period P_i (during which the step-in connectivity remains the same) is separated with vertical dashed lines to its previous and/or following counterpart. For the sake of brevity and clarity, the temporal instance of a node n_j which is active during the period $P_i = [t_i, t_{i+1}[$

[6]To simplify, in this example, an edge becomes unavailable if it is partially covered by the obstacle of if one on its extremal nodes is partially covered. In the RH-DRM framework, the occupancy grid mapping is to be used to determine the availability of nodes and edges, as was described in Section 4.4.2.2

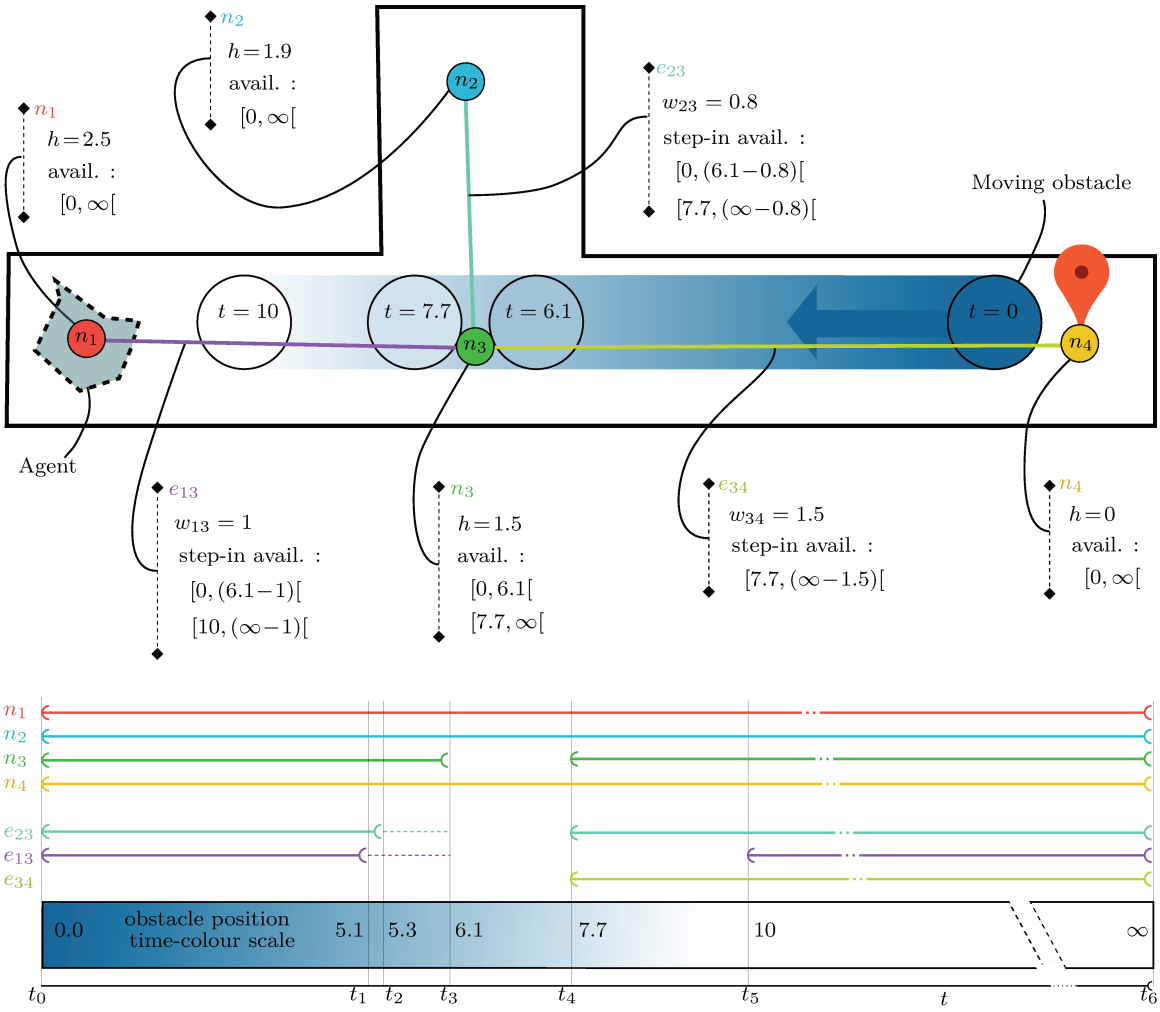


Figure 4.5.2: Illustrative example showing a moving obstacle (a disk) moving from right to left in a narrow corridor. An undirected weighted graph composed of 4 nodes $(n_i)_{i\in[\![1..4]\!]}$ and 3 edges (e_{13}, e_{23} and e_{34}) is superimposed onto the map. The trajectory of the moving obstacle is predicted over a 10 seconds time horizon, which allows to predict nodes and edges availability and step-in availability. In the anticipation, n_1, n_2 and n_4 are always free and have a heuristic value of $h = 2.5s, 1.9s, 0.0s$, respectively. n_3, whose $h = 1.5s$, is occupied by the obstacle between $t_3 = 6.1s$ and $t_4 = 7.7s$. Edge e_{34} is occupied by the obstacle between $t_0 = 0.0s$ and $t_4 = 7.7s$. Its weight is $w_{34} = 1.5s$. Edge e_{23} is occupied by the obstacle between $t_3 = 6.1s$ and $t_4 = 7.7s$. Its weight is $w_{23} = 0.8s$. Edge e_{13} is occupied by the obstacle between $t_3 = 6.1s$ and $t_5 = 10s$, which is the end of the time horizon. Its weight is $w_{34} = 1.0s$. The corresponding interval graph availability intervals are represented at the bottom of the figure along with the time-colour scale.

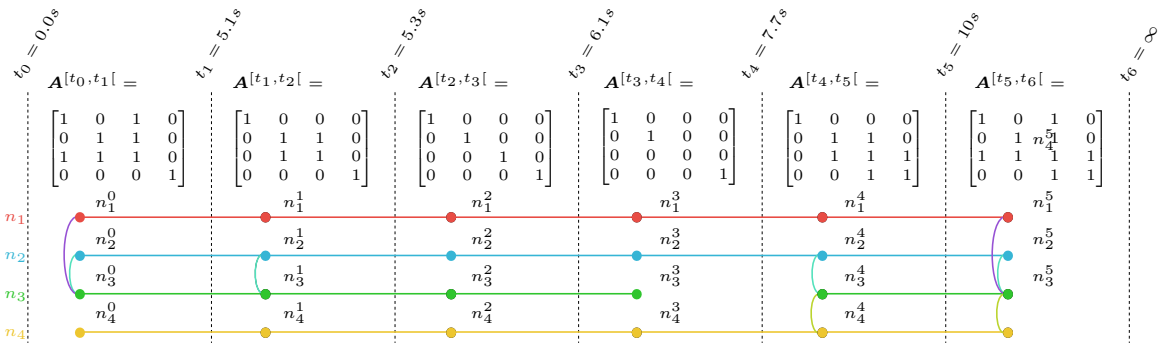


Figure 4.5.3: Interval-graphlets and modified slice graph representation of a step-in interval graph. Note that although the diagonal terms of $\boldsymbol{A}^{[t_5,t_6[}$ are 1, no horizontal connection is drawn on the right of the corresponding NTIs (that is because $t_6 = \infty$).

will be denoted n_j^i. The solid curves linking vertically-aligned NTIs (*e.g. linking* n_j^i *to* n_k^i) indicate which edges of the graph can be stepped in (*i.e.* if the edges can begin to be traversed) during the corresponding period. A horizontal solid line between two time-consecutive node temporal instances n_j^i and n_j^{i+1} denotes the possibility to stay within the configuration associated to node n_j during $[t_i, t_{i+2}[$. The absence of horizontal segment on the right of a NTI indicates that the agent cannot stay after its corresponding time-interval ends (*i.e.* it can still remain there during $[t_i, t_{i+1}[$).

For example, the configurations associated to all nodes may be stayed in during $[t_0, t_2[$ (horizontal lines). From n_1^0, the agent could start the traversal of e_{13} at any time during $[t_0, t_1[$ (vertical curve from the n_1^0 to n_3^0). The same applies to n_2^0 with the traversal of e_{23}. From n_3^0 the agent could start the traversal of e_{13} or e_{23}. However, edge e_{34} could not be stepped in during that period, because it is occupied by the obstacle.

Fifteen clones of the slice graph of Fig. 4.5.3 are displayed on Fig. 4.5.4 and Fig. 4.5.5 to illustrate the incremental exploration performed by Algo. 4. The fifteen algorithm steps which are represented here begin at line 9 and end when the break condition of line 12 is triggered, meaning that the foremost journey was found.

On these figures, the NTIs are the same as the ones of Fig. 4.5.3, but are represented

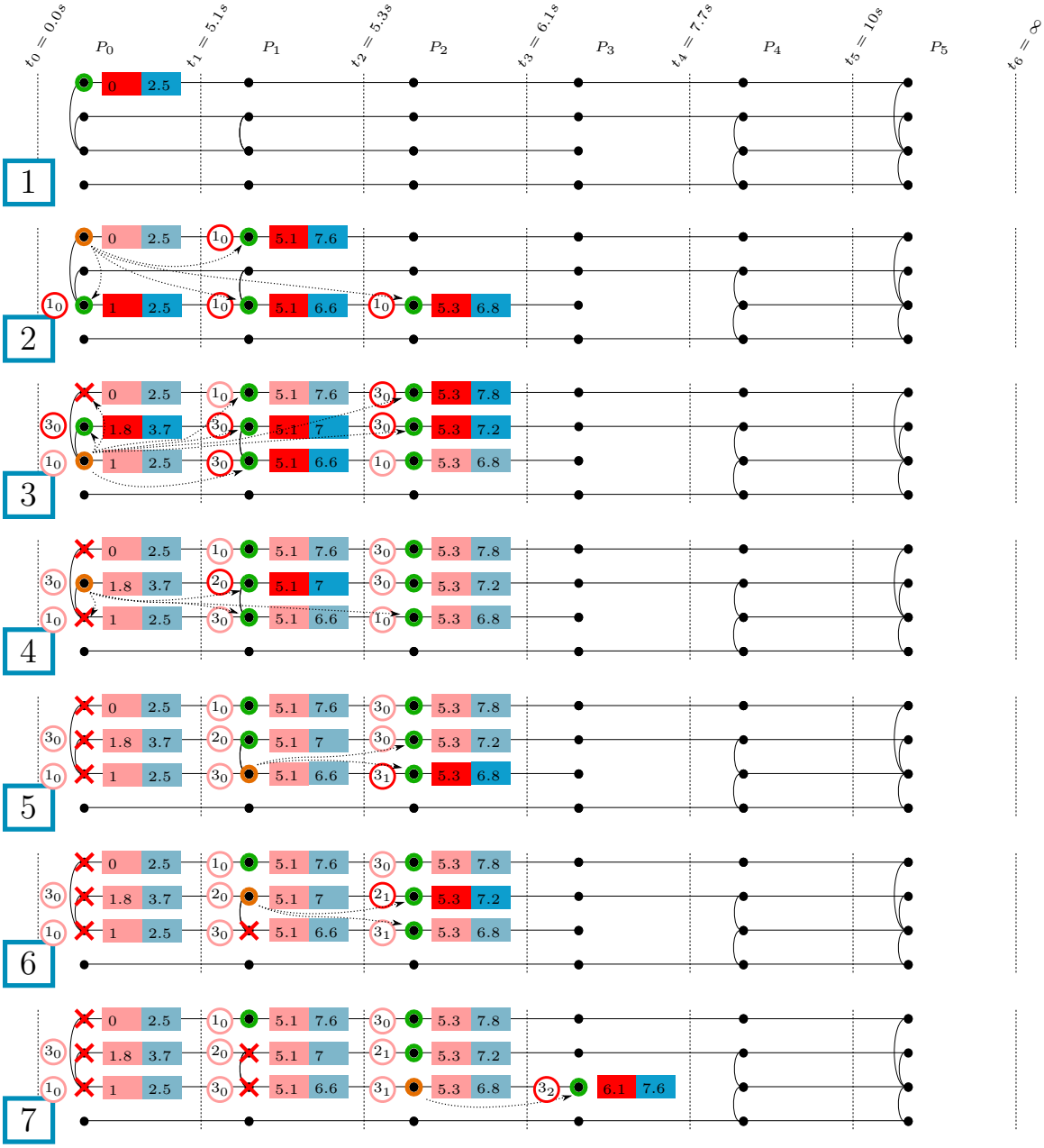


Figure 4.5.4: RH-DRM algorithm explanations (step 1-7).

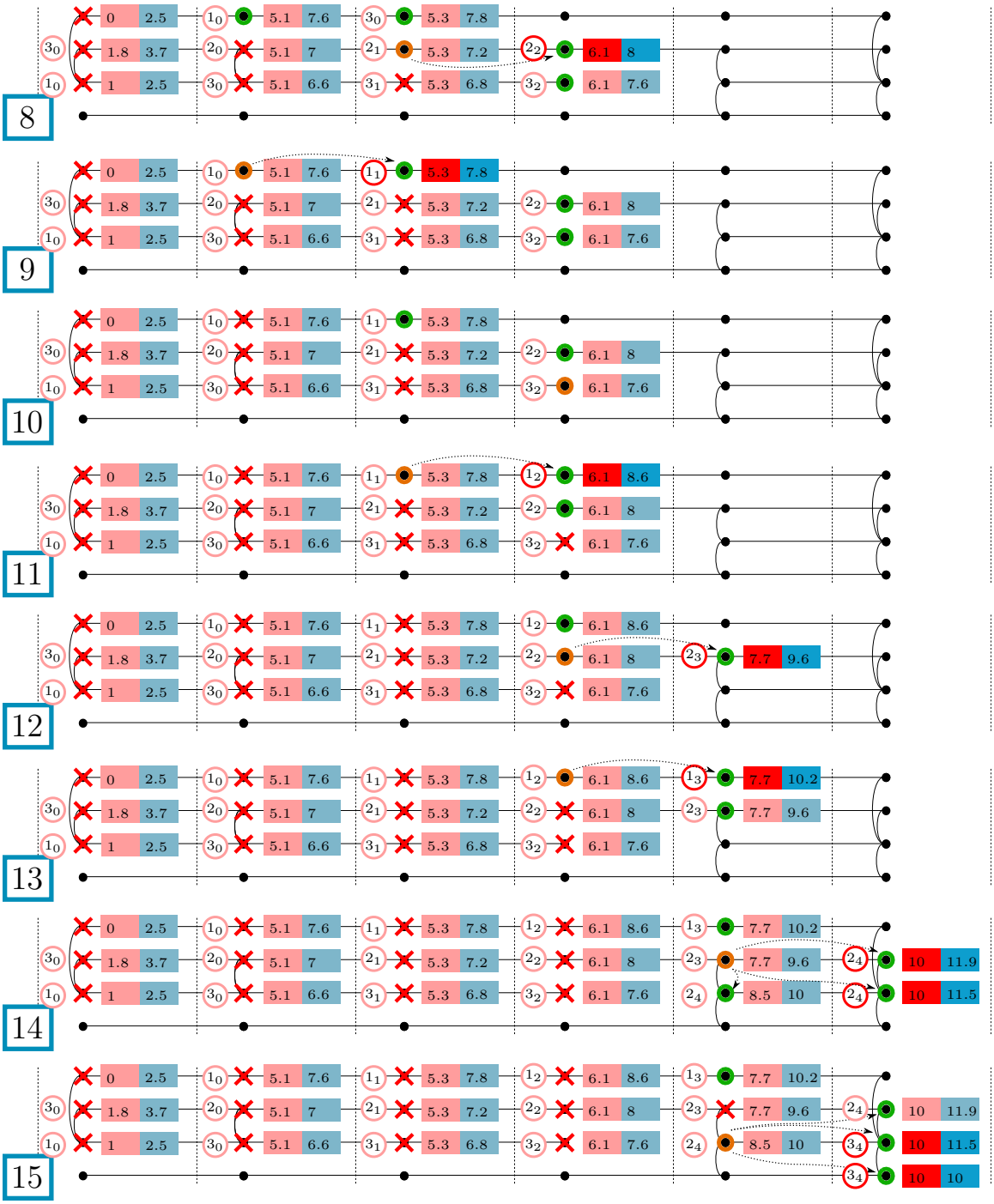


Figure 4.5.5: RH-DRM algorithm explanations (step 8-15).

with black dots •. The red and blue frames 5.1 7.6 contain the g and the $g+h$ values of the NTI situated on the left. If a NTI is circled in green, it belongs to the *open* list. An orange circling indicates which NTI was selected by `pop` function and is currently being expanded. A red cross over a NTI indicates that it belongs to the *closed* list. The circled numbers on the left of the NTIs 1_0 point to the NTI which is the best parent found so far. The first digit corresponds to the reference (*i.e.* the *id*) of the best parent, and the subscript denotes the interval reference ($P_i = [t_i, t_{i+1}[$ is the ith interval or period). *e.g.* 1_0 points to the NTI of the first node n_1 during the first period P_0. During expansion steps (step 2 to 15), the dashed arrows ⋯➤ indicate the neighbouring NTIs that the expansion of the popped NTI reaches.

- Step 1: Insertion of n_1^0 which is reached at time $t = 0$ into *open*.
- Step 2: Expansion of the only NTI contained in *open* : n_1^0. The step-in connectivity of this NTI indicates that it can remain in the same place, or travel towards the 3^{rd} node. The NTI is first expanded towards n_3 instances that can be reached by starting the journey in n_1 between $t = t_0 = 0$ and $t = t_1 - \epsilon < 5.1$, given that the trip duration lasts 1 second. The first arrival in n_3 will occur at time $t = g + w_{13} = 1s$ while there won't be any arrival from $t = u + w_{13} = 6.1s$ onwards. Therefore, the temporal instances n_3^0, n_3^1 and n_3^2 are updated and added to *open*. n_1^0 is then expanded towards its time-successor n_1^1 (dashed arrow going to the right). n_1^1 is added to the *open* list with $g = t_1 = 5.1s$.
- Step 3: The most promising node which is popped is n_3^0, because its $g + h$ score is the smallest of the NTIs contained in the *open* list. The NTI is step-in-connected to n_1, n_2 and to its successor. The relevant expansions and updates are performed.
- Step 4-13: The same pattern is executed with little advances on the journey. This is explained by the fact that the agent needs to wait for the obstacle to free n_3 up, in order to be able to move again.

- Step 14: Once the obstacle has left n_3 ($t = 7.7s$), the agent can safely travel from n_2 to n_3. The expansion of the NTI of n_2 for the period $P_4 = [7.7, 10[$ is performed. n_3 can be reached as early as $t = 8.5s$.
- Step 15: The expansion of n_3^4 is performed, and allows to finally reach n_4 at $t = 10s$. This NTI will be popped in the next loop because it is contained in *open* and scores the smallest value of $g + h$. The the stop condition will be triggered.

Given the problem inputs, the foremost journey can be determined by cascading down the parentage of the first NTI associated to n_4 which is popped : n_4^5. n_4 is reached at $t = 10s$, and its best parent is n_3^4. Arriving in n_4 at $t = 10s$ implies leaving n_3 at $t = 10 - w_{23} = 10 - 1.5 = 8.5s$. The best parent of n_3^4 is n_2^4. Therefore, the agent should leave n_2 at $t = 8.5 - w_{23} = 8.5 - 0.8 = 7.7$. The best parent of n_2^4 is n_2^3, whose best parent is n_2^2, etc.. until n_2^0. Therefore, the agent will remain in n_2 from $t = 1.8$ to $t = 7.7s$. The best parent of n_2^0 is n_3^0. The agent should therefore leave n_3 at $t = 1.8 - w_{23} = 1.8 - 0.8 = 1s$. The best parent of n_3^0 is n_1^0, which must be left at $t = 1 - w_{13} = 1 - 1 = 0s$. The journey can therefore be synthesised in the following steps :

1. Leave n_1 at $t = 0s$ to arrive in n_3 at $t = 1s$.
2. Leave n_3 at $t = 1s$ to arrive in n_2 at $t = 1.8s$.
3. Stay in n_2 until $t = 7.7s$.
4. Leave n_2 at $t = 7.7s$ to arrive in n_3 at $t = 8.5s$.
5. Leave n_3 at $t = 8.5s$ to arrive in n_4 at $t = 10s$.

4.6 Intermediary Conclusion

This chapter introduced the Receding Horizon Dynamic RoadMaps (RH-DRM), a motion planning strategy that combines the framework of dynamic roadmaps with a

novel temporal network search method. Collision-free trajectories are found efficiently within environments that may move faster than the agent itself. The computed trajectories adapt to the environment's motions, and thus the impact and intrusiveness of the solution is small on the environment's flow. The strategy is applicable for mobile systems and serial systems, and handles even complex geometries online, because the costly collision computation part is performed offline. The dynamic roadmap framework is used to discreetly map the obstacle space into the configuration space of the agent. Doing so allows determining efficiently the future topology of the connectivity graph (the roadmap), by translating an anticipation of the moving obstacles trajectories. The foremost journey, *i.e.* the earliest arrival journey, is found within this changing graph thanks to a novel temporal network search algorithm inspired for the A* algorithm.

To be implemented on real systems, the strategy requires the use of an efficient predictor of the moving obstacles trajectories. If the other obstacles are robots, and assuming a hierarchical organisation is chosen, where some robots have a path planning priority over others, the anticipation of these top priority robot trajectories may be shared, allowing for an exact prediction of space occupancy. If the moving obstacles are humans, an efficient and reliable predictor of their trajectories must be put in place. Predicting human motions is an active research domain [150–159]. To generate socially acceptable, yet efficient robot motions, the field of human-aware navigation [160, 161] studies and predicts the joint influence of humans and robots over the motions they produce. In one of this field-related approaches [162], the robot trajectory generation takes the form of an optimisation problem over time elastic bands [163, 164] which account for the robot trajectory and the humans'. Doing so, the joint influence of the human and robot trajectories are predicted by stretching their respective elastic bands until they all abide by constraints on safety and social rules. By jointly predicting (planning) the trajectory that a moving human will take in the presence of a robot, and the one of the robot, deadlocks can be avoided very elegantly.

The optimality of the prediction depends directly on the quality of the prediction

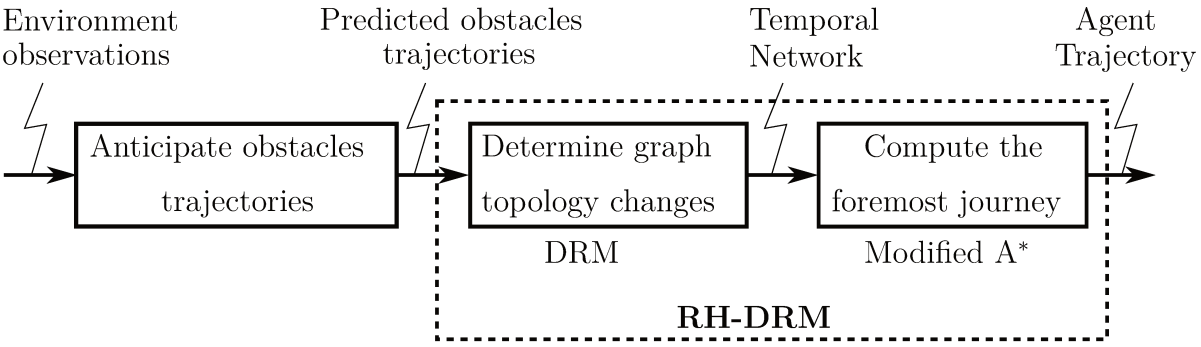


Figure 4.6.1: RH-DRM bloc diagram.

and on the duration of the prediction horizon.

To the author's knowledge, the modified A* algorithm presented here is the first that solves the problem of the foremost journey in that manner. The main challenges that were tackled are the changing nature of the graph, whose edges and nodes may be unavailable for periods of time, and the fact that edges traversals are not instantaneous. The overall RH-DRM framework is adaptable to complex geometries, serial or mobile systems, thanks to the underlying DRM strategy.

On the other hand, the RH-DRM suffers from some limitations. In the strategy described therein, the agent can only move at the speed imposed by the edges weights or stop, which can be somewhat limiting. A solution to this restriction could be to duplicate each edge of the static graph and set a different weight to each of the clones.

Another limitation of the scheme is related to the fact that the duration of the receding horizon is directly correlated to the computational cost. The further in time the prediction of the trajectories goes, the more discardings of nodes and edges will occur. Each discarding corresponds to a date where the step-in connectivity changes (unless, as is often the case, several discardings occur at the same date). The number of step-in connectivity changes (the number of slices used to represent the temporal network) is itself roughly proportional to the computational load of the algorithm.

The variations of obstacle-freedom of a node or an edge have no influence on the

foremost journey of the agent if the latter is not able to reach it before they occur. Therefore, the variations of obstacle-freedom of all nodes and edges only have to be taken into account from the moment these elements could possibly be reached by the agent. In particular, all nodes and edges that are situated out of the time-horizon-reach of the agent can be completely ignored during the temporal network construction. The voxels that are only concerned by these elements don't even need to be checked for collision with the predicted trajectories of the obstacles. The implications of this are two fold.

- The collision checks computational load can be greatly reduced, by monitoring only the part of the environment that could be concerned by collisions between the agent and the moving obstacles.
- The temporal network size (and therefore the computational load of the search within it) can be reduced dramatically.

To determine the nodes and edges that are concerned by the agent during the time horizon, a modified Dijkstra algorithm can be used on the original static graph (*i.e.* the unchanged graph, with all its nodes and edges). This version of the algorithm can be seen on Algo. 6. The principle of the algorithm is to radiate outwards from the node which is currently occupied by the agent (sStart), until the cost of the shortest path to a node exceeds the cost $t_{horizon}$. A list (*closed*) containing the references to all the nodes that where expanded during the contagion phase is then returned by the algorithm.

A simulation showing the RH-DRM algorithm in action can be seen on Fig. 4.6.2. On this figure, the agent (blue shape) is to move within a closed space in which four obstacles (red shapes) are moving. The planned trajectory (position and orientation of the agent in the plane) is depicted with a red path interspersed by red arrows, showing the orientation of the agent. The previous state of the agent, of each obstacle and of the planned trajectory is overlapped in lighter colours onto each step state. In this simulation, the anticipation of the obstacles trajectories is performed thanks to

a linear extrapolation based on the current configuration of the obstacle and the one from the previous state. The obstacles and agent have similar velocity capabilities.

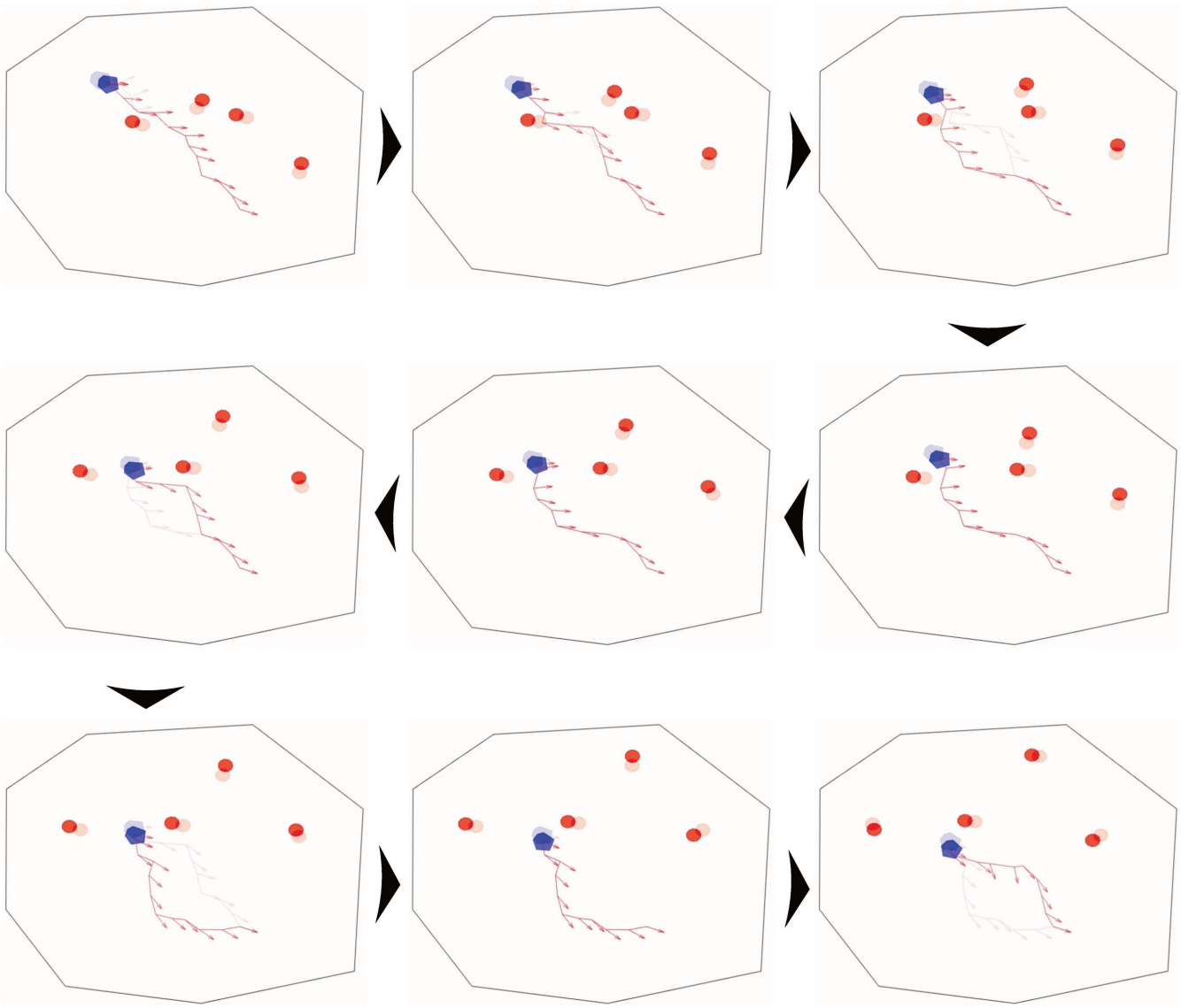
input :

- $\mathcal{G}(\mathcal{N}, \mathcal{E})$ `// a graph (`$\mathcal{G}$`) comprised of nodes (`$\mathcal{N}$`) and edges (`$\mathcal{E}$`)`
- sStart `// The reference to the starting node`
- $t_{horizon}$ `// The duration of the prediction horizon`

output: *closed* `// A list of nodes that can be reached during` $t_{horizon}$

```
1  Procedure ModifiedDijkstra(G(N,E), sStart, t_horizon)
2      forall s ∈ N do s.g ← ∞ ;
3      sStart.g ← 0 ;
4      open ← ∅, closed ← ∅ ;
5      add(sStart,open) ;
6      loop
7          s ← pop(open) ;
8          if g.s > t_horizon then break loop ;
9          else if s = ∅ then return ∅ ;
10         foreach n ∈ neighbours(s,E), n ∉ closed do
11             costToNThroughS ← s.g + cost(s,n,E) ;
12             if costToNThroughS < n.g then
13                 n.g ← costToNThroughS ;
14                 if n ∉ open then add(n,open) ;
15             end
16         end
17         add(s,closed)
18     end
19     return closed;
20 end
21 Procedure pop(open)
22     s ← arg min_{c∈open}(c.g);
23     remove(s,open);
24     return s;
25 end
```

Algorithm 6: Modified Dijkstra's algorithm aiming at collecting all nodes that can be reached by the agent within a prediction horizon $t_{horizon}$.



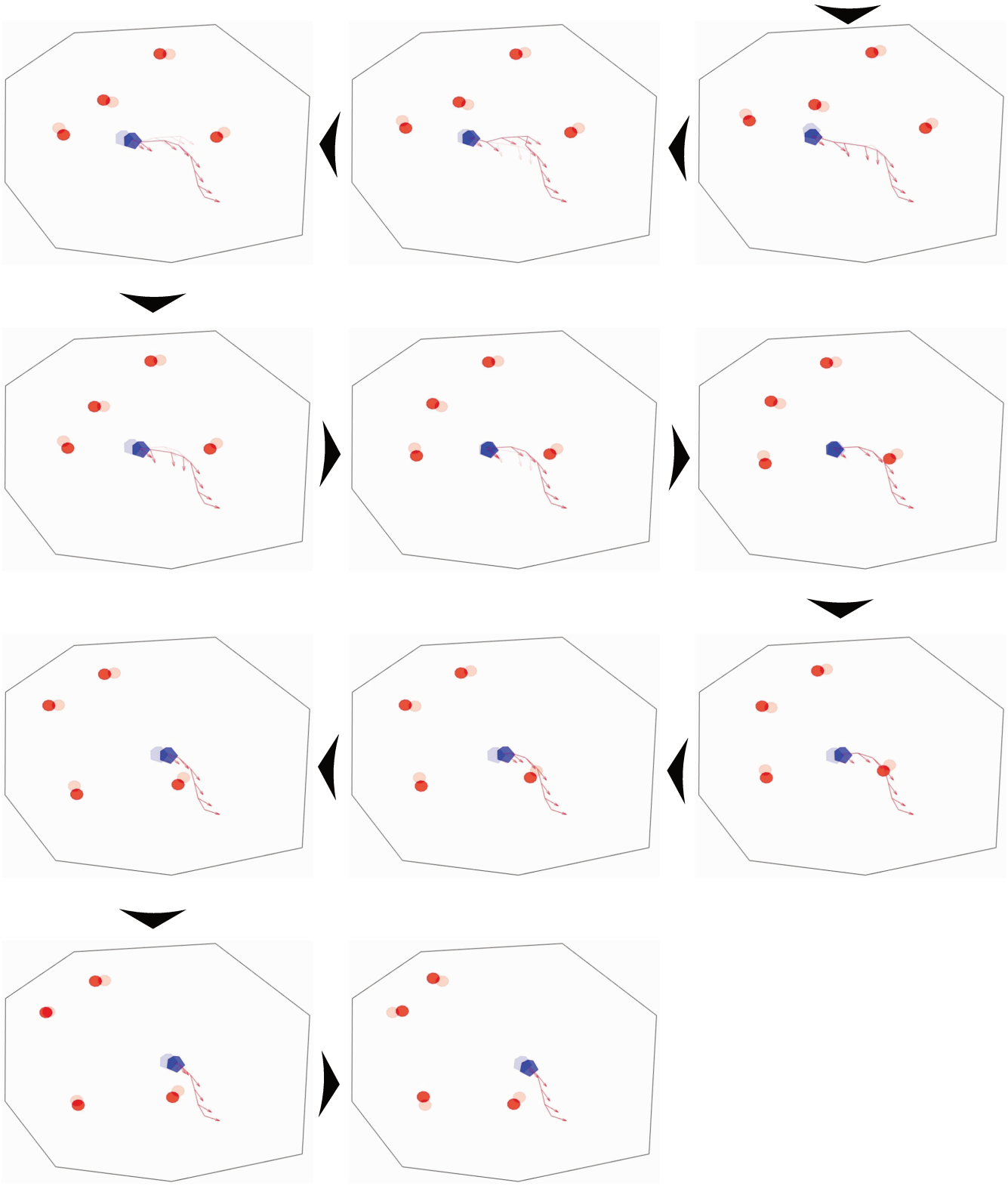


Figure 4.6.2: Motion planning for a 3-DOFs mobile robot using RH-DRM. The agent is depicted in blue, the (static) boundaries in black, and the moving obstacles in red, and the planned trajectory in red. For each step, the previous state of the problem is lightly shaded over the current one.

Conclusions

Whilst traditional robotised cells can almost always be completely defined and planned before they are ever constructed or used, the unstructured and dynamic environment that is a human populated shop-floor makes the process of thoroughly listing every possible situation tedious or even impossible. The changing nature of the environment prevents the use of basic logical rules and requires advanced features. Having humans in the vicinity places versatility and adaptability on center stage. Autonomous solutions, that are able to accommodate with a changing environment, have to be developed in this context.

One of the first responsibility of these autonomous systems is to choose suitable postures for the tasks that are ordered to them. The use of redundant systems is justified here because the flexibility redundancy offers on reachability is very advantageous. The alternative options offered by redundancy on the postures used are essential in a cluttered and changing environment. However, from a computational and mathematical perspective, redundancy also complicates the inverse modelling step needed to determine these postures. This complication is the first industrial lock that was addressed in this report. A focus was made on positional tasks, *i.e.* tasks requiring the end-effector to be positioned in a static location, because they constitute a large part of the industrial tasks ordered to robots. A characterisation of positional redundancy was made through a parameterisation of the self-displacement space of redundant systems. In the examples derived in the thesis, a position in this redundancy space could be specified along with a positional task to provide a unique, analytically-found articular configuration. The advantages of solving redundancy for positional tasks at the position level, instead of doing it using differential schemes, were exposed in the conclusion of Chapter 1. Among these advantages is the fact that

boundaries of the solution space are more easily apprehended, and provides with a clearer idea of the extent of the available options. The benefits of using redundancy spaces for positional redundancy resolution were also exposed for the optimisation of a posture dependant criteria. No approximation is made if the inverse modelling step is analytic, and the computation is also much faster. The dimension of the search space of the optimisation problem is reduced to the number of redundant parameters that direct the redundancy space, instead of having a dimension equal to the number of degrees of freedom of the robot.

A suitable posture for these redundant system does not only mean a posture that ensures reachability. Industrial processes have accuracy requirements that need to be met. When industrial processes involve a physical interaction with the environment, deformations within the kinematic structure of the robot can occur and cause a misplacement of the end-effector. A characterisation of the misplacement caused by a force acting on the end-effector of serial systems was performed in Chapter 2, and was shown to be significantly influenced by the posture used. A strategy using the framework of redundancy spaces was exposed to find postures that fulfilled the geometric task while limiting end-effector misplacements.

Another requirement for the fulfilment of a positional task requiring physical interaction with the environment (such as machining operations, or assembly processes), is the assurance that the system is able to counter the contact forces of the interaction. A second type of posture dependant criterion which accounts for the force capacity of serial systems, in a chosen configuration, for a given force applied somewhere on the end-effector was described in Chapter 3. The scope of this index differs from the one of force ellipsoids, force polytopes of dynamic capability equations because it focuses on a single force definition, and was designed for redundancy resolution. A strategy using the framework of redundancy spaces was again exposed to find postures that ensured the capacity of the robot to produce a given force.

Exemplary use case involving a LBR iiwa for drilling tasks were described to illus-

trate the use of these two posture dependant criteria for positional redundancy resolution. The tools developed therein were shown to be interesting for the convenience they brings to optimisation schemes. Additionally the realm of possibilities offered by redundancy was explored quickly and efficiently. This exposed the im-possibility to perform a positional task while fulfilling the accuracy requirements more thoroughly and globally than schemes that would be based on redundancy resolution at a differential level.

Instead of schemes seeking for the "best posture" with regards to one several criteria, the industrial locks related to deformational behaviour and force capacity were described as constraint satisfaction problems. This strategy allows to make multi criteria decisions without worrying about a Pareto dilemma while leaving room for other posture dependant criteria to come within the scheme. An important feature of these criteria is their simplicity. Additionally, a special effort was made to produce criteria that are closely related to how the industrial needs are formulated. Instead of optimising rigidity, the optimisation is made of the misplacement of the end-effector, which is directly homogeneous and comparable to mechanical tolerances. The force capacity index (FCI) can be thought of as a safety coefficient on the assurance that a robot will be able to sustain a force. The criteria is intuitive, easily understood which likewise makes it a good candidate for industrial implementations.

The last industrial lock that was addressed is the one of performing safe motions within a human populated environment. The unstructured nature of this environment makes the problem very complicated. In this thesis work, the intuition was to use and adapt the well-known sampling-based method that is the Probabilistic RoadMap (PRM) to the dynamic and changing nature of the environment. The novel motion planning strategy is named Receding Horizon Dynamic RoadMaps (RH-DRM). PRM-based methods have the advantage of being efficient, and adaptable to serial and mobile robot problems alike, because the path planning is performed in configuration space. An adaptation of PRM, called Dynamic RoadMaps (DRM), was taken as basis for the final scheme. The main difference with DRM consists in using the anticipation

of the trajectories of the obstacles instead of just using their spatial occupancy at a single instant (the current time). The temporal paradigm that was added to the problem led to the development of a novel adaptation of the A* algorithm that finds an earliest arrival journey within an anticipated graph topology where edges traversal is not instantaneous. A type of temporal network, here named step-in interval graph, was defined to formalise and solve the problem at hand. To the author's knowledge, this algorithm is the first that solves this problem in that manner.

RH-DRM retains the advantages of PRM and DRM. It is fast, adaptable to any kind of system (planning is done in configuration space), even those with many DOFs. Additionally, the algorithm provides the foremost collision-free journey, integrates anticipation, which helps avoiding deadlocks, collisions, ensures efficiency of the journey, and allows planning even for systems that may be slower than the dynamic obstacles. These last advantages are completely dependant on the quality of the obstacles trajectories prediction.

Some improvements or perspectives could be explored for the industrial implementation of the deformational behaviour criteria described in this thesis.

These indices are based on the assumption that the deformations only happen within the articulations of the robots. It could be interesting to explore the influence of the links of the robot, as well as the deformation of the end-effector on the deformational behaviour of the system.

Additionally, a very straightforward interaction force definition was used in the chapter related to the deformational behaviour, which didn't precisely reflect the complexity of the real interaction happening between a drilling end-effector, mounted on a deformed robot and entering an aluminium plate while being clamped against the plate. The forces issued from the deformation of the drill, as well as from the friction of the clamp onto the plate act together to improve the deformational behaviour of the robot by opposing the misplacement of the end-effector. It would be interesting to look into the effects of these elements onto the deformational behaviour of the

robot and machining quality.

The sensibility of the posture-dependant criteria described in this thesis, with regard to the interaction force between the end-effector and the environment could be explored. Indeed, what would be the influence of an error made on the estimation of the intensity or direction of the spatial force related to the interaction? The knowledge of this sensibility could be a safeguard to slightly wavering interaction forces or for inaccurate estimations. An equivalent, but somehow more intuitive result would be the knowledge of the worst case deformational behaviour and/or force capacity indices, for input forces that are relatively "close" to the estimated one. Note that this sensibility is somehow equivalent to the one related to the positional error made on the estimated force application point.

The RH-DRM strategy is directly dependant on the obstacles trajectory predictions. Therefore, an implementation of this algorithm would require to develop the anticipation part of the problem. This aspect is to be included in further developments. The sensing part of the problem would be a major lock from a safety-related perspective, if the solution is ever to be implemented in an industrial context.

Appendix 1 : Iiwa Direct Geometric Modelling

The expressions of the elementary homogeneous matrices derived from the DH parametrisation of the system (see 1.3.1) are :

$$
{}^0T_1 = \begin{pmatrix} c_{q1} & -s_{q1} & 0 & 0 \\ s_{q1} & c_{q1} & 0 & 0 \\ 0 & 0 & 1 & l_{bs} \\ 0 & 0 & 0 & 1 \end{pmatrix} \qquad {}^1T_2 = \begin{pmatrix} c_{q2} & -s_{q2} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -s_{q2} & -c_{q2} & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}
$$

$$
{}^2T_3 = \begin{pmatrix} c_{q3} & -s_{q3} & 0 & 0 \\ 0 & 0 & -1 & -l_{se} \\ s_{q3} & c_{q3} & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \qquad {}^3T_4 = \begin{pmatrix} c_{q4} & -s_{q4} & 0 & 0 \\ 0 & 0 & -1 & 0 \\ s_{q4} & c_{q4} & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}
$$

$$
{}^4T_5 = \begin{pmatrix} c_{q5} & -s_{q5} & 0 & 0 \\ 0 & 0 & 1 & l_{ew} \\ -s_{q5} & -c_{q5} & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \qquad {}^5T_6 = \begin{pmatrix} c_{q6} & -s_{q6} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -s_{q6} & -c_{q6} & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}
$$

$$
{}^6T_7 = \begin{pmatrix} c_{q7} & -s_{q7} & 0 & 0 \\ 0 & 0 & -1 & -l_{wt} \\ s_{q7} & c_{q7} & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{4.3}
$$

The components of ${}^0\mathbf{T}_7$ computed from the previous matrices multiplication are found in Eq. (4.4). We adopt Matlab matrix notation, *i.e.* for a $q \times p$ matrix A, for any

$i \in [\![1..q]\!]$ and any $j \in [\![1..p]\!]$, $A(i,j)$ is the term situated at the i^{th} row and j^{th} column of A.

$$
\begin{aligned}
{}^0\mathbf{T}_7(1,1) =& s_{q7}(s_{q5}(c_{q4}(s_{q1}s_{q3} - c_{q1}c_{q2}c_{q3}) - c_{q1}s_{q2}s_{q4}) - c_{q5}(c_{q3}s_{q1} + c_{q1}c_{q2}s_{q3}))- \\
& c_{q7}(s_{q6}(s_{q4}(s_{q1}s_{q3} - c_{q1}c_{q2}c_{q3}) + c_{q1}c_{q4}s_{q2})+ \\
& c_{q6}(s_{q5}(c_{q3}s_{q1} + c_{q1}c_{q2}s_{q3}) + c_{q5}(c_{q4}(s_{q1}s_{q3} - c_{q1}c_{q2}c_{q3}) - c_{q1}s_{q2}s_{q4})))
\end{aligned}
$$

$$
\begin{aligned}
{}^0\mathbf{T}_7(2,1) =& c_{q7}(s_{q6}(s_{q4}(c_{q1}s_{q3} + c_{q2}c_{q3}s_{q1}) - c_{q4}s_{q1}s_{q2}) + c_{q6}(s_{q5}(c_{q1}c_{q3} - c_{q2}s_{q1}s_{q3})+ \\
& c_{q5}(c_{q4}(c_{q1}s_{q3} + c_{q2}c_{q3}s_{q1}) + s_{q1}s_{q2}s_{q4})))- \\
& s_{q7}(s_{q5}(c_{q4}(c_{q1}s_{q3} + c_{q2}c_{q3}s_{q1}) + s_{q1}s_{q2}s_{q4}) - c_{q5}(c_{q1}c_{q3} - c_{q2}s_{q1}s_{q3}))
\end{aligned}
$$

$$
\begin{aligned}
{}^0\mathbf{T}_7(3,1) =& -s_{q7}(s_{q5}(c_{q2}s_{q4} - c_{q3}c_{q4}s_{q2}) - c_{q5}s_{q2}s_{q3}) - c_{q7}(s_{q6}(c_{q2}c_{q4} + c_{q3}s_{q2}s_{q4})- \\
& c_{q6}(c_{q5}(c_{q2}s_{q4} - c_{q3}c_{q4}s_{q2}) + s_{q2}s_{q3}s_{q5}))
\end{aligned}
$$

$$
\begin{aligned}
{}^0\mathbf{T}_7(1,2) =& s_{q7}(s_{q6}(s_{q4}(s_{q1}s_{q3} - c_{q1}c_{q2}c_{q3}) + c_{q1}c_{q4}s_{q2}) + c_{q6}(s_{q5}(c_{q3}s_{q1} + c_{q1}c_{q2}s_{q3})+ \\
& c_{q5}(c_{q4}(s_{q1}s_{q3} - c_{q1}c_{q2}c_{q3}) - c_{q1}s_{q2}s_{q4})))+ \\
& c_{q7}(s_{q5}(c_{q4}(s_{q1}s_{q3} - c_{q1}c_{q2}c_{q3}) - c_{q1}s_{q2}s_{q4}) - c_{q5}(c_{q3}s_{q1} + c_{q1}c_{q2}s_{q3}))
\end{aligned}
$$

$$
\begin{aligned}
{}^0\mathbf{T}_7(2,2) =& -s_{q7}(s_{q6}(s_{q4}(c_{q1}s_{q3} + c_{q2}c_{q3}s_{q1}) - c_{q4}s_{q1}s_{q2}) + c_{q6}(s_{q5}(c_{q1}c_{q3} - c_{q2}s_{q1}s_{q3})+ \\
& c_{q5}(c_{q4}(c_{q1}s_{q3} + c_{q2}c_{q3}s_{q1}) + s_{q1}s_{q2}s_{q4})))- \\
& c_{q7}(s_{q5}(c_{q4}(c_{q1}s_{q3} + c_{q2}c_{q3}s_{q1}) + s_{q1}s_{q2}s_{q4}) - c_{q5}(c_{q1}c_{q3} - c_{q2}s_{q1}s_{q3}))
\end{aligned}
$$

$$
\begin{aligned}
{}^0\mathbf{T}_7(3,2) =& s_{q7}(s_{q6}(c_{q2}c_{q4} + c_{q3}s_{q2}s_{q4}) - c_{q6}(c_{q5}(c_{q2}s_{q4} - c_{q3}c_{q4}s_{q2}) + s_{q2}s_{q3}s_{q5}))- \\
& c_{q7}(s_{q5}(c_{q2}s_{q4} - c_{q3}c_{q4}s_{q2}) - c_{q5}s_{q2}s_{q3})
\end{aligned}
$$

$$
\begin{aligned}
{}^0\mathbf{T}_7(1,3) =& c_{q6}(s_{q4}(s_{q1}s_{q3} - c_{q1}c_{q2}c_{q3}) + c_{q1}c_{q4}s_{q2}) - s_{q6}(s_{q5}(c_{q3}s_{q1} + c_{q1}c_{q2}s_{q3})+ \\
& c_{q5}(c_{q4}(s_{q1}s_{q3} - c_{q1}c_{q2}c_{q3}) - c_{q1}s_{q2}s_{q4}))
\end{aligned}
$$

$$
\begin{aligned}
{}^0\mathbf{T}_7(2,3) =& s_{q6}(s_{q5}(c_{q1}c_{q3} - c_{q2}s_{q1}s_{q3}) + c_{q5}(c_{q4}(c_{q1}s_{q3} + c_{q2}c_{q3}s_{q1}) + s_{q1}s_{q2}s_{q4}))- \\
& c_{q6}(s_{q4}(c_{q1}s_{q3} + c_{q2}c_{q3}s_{q1}) - c_{q4}s_{q1}s_{q2})
\end{aligned}
$$

$$
{}^0\mathbf{T}_7(3,3) = s_{q6}(c_{q5}(c_{q2}s_{q4} - c_{q3}c_{q4}s_{q2}) + s_{q2}s_{q3}s_{q5}) + c_{q6}(c_{q2}c_{q4} + c_{q3}s_{q2}s_{q4})
$$

$$
\begin{aligned}
{}^0\mathbf{T}_7(1,4) =& c_{q6}lwt(s_{q4}(s_{q1}s_{q3} - c_{q1}c_{q2}c_{q3}) + c_{q1}c_{q4}s_{q2}) - lwts_{q6}(s_{q5}(c_{q3}s_{q1} + c_{q1}c_{q2}s_{q3})+ \\
& c_{q5}(c_{q4}(s_{q1}s_{q3} - c_{q1}c_{q2}c_{q3}) - c_{q1}s_{q2}s_{q4})) + lews_{q4}(s_{q1}s_{q3} - c_{q1}c_{q2}c_{q3})+ \\
& c_{q1}lses_{q2} + c_{q1}c_{q4}lews_{q2}
\end{aligned}
$$

$$
\begin{aligned}
{}^0\mathbf{T}_7(2,4) =& lwts_{q6}(s_{q5}(c_{q1}c_{q3} - c_{q2}s_{q1}s_{q3}) + c_{q5}(c_{q4}(c_{q1}s_{q3} + c_{q2}c_{q3}s_{q1}) + s_{q1}s_{q2}s_{q4}))- \\
& lews_{q4}(c_{q1}s_{q3} + c_{q2}c_{q3}s_{q1}) - c_{q6}lwt(s_{q4}(c_{q1}s_{q3} + c_{q2}c_{q3}s_{q1}) - c_{q4}s_{q1}s_{q2})+ \\
& lses_{q1}s_{q2} + c_{q4}lews_{q1}s_{q2}
\end{aligned}
$$

$$
\begin{aligned}
{}^0\mathbf{T}_7(3,4) =& lbs + c_{q2}lse + lwts_{q6}(c_{q5}(c_{q2}s_{q4} - c_{q3}c_{q4}s_{q2}) + s_{q2}s_{q3}s_{q5}) + c_{q6}lwt(c_{q2}c_{q4}+ \\
& c_{q3}s_{q2}s_{q4}) + c_{q2}c_{q4}lew + c_{q3}lews_{q2}s_{q4}
\end{aligned}
$$

(4.4)

Appendix 2 : Inverse geometric model of the LBR IIWA parameterised by the elbow angle - a demonstration

The demonstration will be decomposed as such :

- Geometric task description under a mathematical form that highlights the spherical-rotary-spherical (S-R-S) structure of a 7 DOFs anthropomorphic manipulators.
- Integration of the elbow angle (β) within this expression.
- Derivation and expression of the non-varying terms of this expression, including q_4.
- β-parameterisation and derivation of the shoulder joint angles (q_1, q_2 and q_3).
- β-parameterisation and derivation of the shoulder joint angles (q_5, q_6 and q_7).

The expressions of the joint angles, with regards to the elbow angle will be distilled throughout the demonstration (Eq. (4.13), Eq. (4.21) and Eq. (4.23)).

Task description with S-R-S structure highlight : Following the S-R-S decomposition described beforehand, the tip position and orientation ${}^0\mathbf{p}_7$ and ${}^0\mathbf{R}_7$ with

regard to the robot base reference frame $\mathcal{F}_0$ can be expressed as such:

$$\begin{aligned} \underbrace{{}^0\mathbf{p}_7}_{\equiv\overrightarrow{\mathrm{BT}}} &= \underbrace{{}^0\mathbf{x}_{bs}}_{\equiv\overrightarrow{\mathrm{BS}}} + {}^0\mathbf{R}_3\Big(\underbrace{{}^3\mathbf{x}_{se}}_{\equiv\overrightarrow{\mathrm{SE}}} + {}^3\mathbf{R}_4\big(\underbrace{{}^4\mathbf{x}_{ew}}_{\equiv\overrightarrow{\mathrm{EW}}} + {}^4\mathbf{R}_7\,\underbrace{{}^7\mathbf{x}_{wt}}_{\equiv\overrightarrow{\mathrm{WT}}}\big)\Big) \\ {}^0\mathbf{R}_7 &= \underbrace{{}^0\mathbf{R}_3}_{shoulder}\ \underbrace{{}^3\mathbf{R}_4}_{elbow}\ \underbrace{{}^4\mathbf{R}_7}_{wrist}. \end{aligned} \tag{4.5}$$

where:

$${}^0\mathbf{x}_{bs} = \begin{bmatrix} 0 \\ 0 \\ l_{bs} \end{bmatrix}, {}^3\mathbf{x}_{se} = \begin{bmatrix} 0 \\ -l_{se} \\ 0 \end{bmatrix}, {}^4\mathbf{x}_{ew} = \begin{bmatrix} 0 \\ l_{ew} \\ 0 \end{bmatrix}, {}^7\mathbf{x}_{wt} = \begin{bmatrix} 0 \\ -l_{wt} \\ 0 \end{bmatrix} \tag{4.6}$$

are derived from the DH parameterisation of the LBR iiwa (see Section 1.3.2.1).

Integration of the elbow angle : Because of the existing redundancy of the elbow angle, the wrist joint input orientation ${}^0\mathbf{R}_4$ can be expressed as

$${}^0\mathbf{R}_4 = \boldsymbol{r_\beta}\,{}^0\mathbf{R}_4^\circ, \tag{4.7}$$

where $\boldsymbol{r_\beta}$ is a rotation matrix operator corresponding to the orientational change resulting from the rotation by the angle β about the shoulder-wrist axis, itself directed by the unitary vector ${}^0\mathbf{u}_{sw} = {}^0\mathbf{x}_{sw}/\|\,{}^0\mathbf{x}_{sw}\|$. Matrix $\boldsymbol{r_\beta}$ is computed thanks to the formula:

$$\begin{aligned} \boldsymbol{r_\beta} &= \boldsymbol{I_3} + \sin(\beta)[{}^0\mathbf{u}_{sw_\times}] + (1-\cos(\beta))[{}^0\mathbf{u}_{sw_\times}]^2 \\ &= \sin(\beta)[{}^0\mathbf{u}_{sw_\times}] - \cos(\beta)[{}^0\mathbf{u}_{sw_\times}]^2 + (\boldsymbol{I_3} + [{}^0\mathbf{u}_{sw_\times}]^2) \end{aligned} \tag{4.8}$$

which derives from the angle axis formalism [6]. In Eq. (4.8), $\boldsymbol{I_3}$ denotes the 3×3 identity matrix, and $[{}^0\mathbf{u}_{sw_\times}]$ denotes the skew-symmetric matrix of vector ${}^0\mathbf{u}_{sw}$.

Now using the fact that the distance from shoulder to wrist is constant whatever the elbow angle :

$$\underbrace{\|\,{}^0\mathbf{u}_{sw}\|}_{cst} = \|\,\underbrace{{}^3\mathbf{x}_{se}}_{cst} + {}^3\mathbf{R}_4\,\underbrace{{}^4\mathbf{x}_{ew}}_{cst}\|, \tag{4.9}$$

it follows that ${}^3\mathbf{R}_4$ is also constant with regard to β. Therefore, $\forall \beta \in \mathbb{R}, {}^3\mathbf{R}_4 = {}^3\mathbf{R}_4^\circ$. Intuitively, looking at the motion of the elbow around the circle displayed on Fig. 1.3.5, it can be intuitively understood that the angle $(\overrightarrow{\mathrm{SE}}, \hat{\overrightarrow{\mathrm{EW}}})$ remains unchanged.

Now looking back at Eq. (4.7), and rearranging it to integrate the shoulder rotation:

$$\begin{aligned} {}^0\mathbf{R}_3\, {}^3\mathbf{R}_4 &= \boldsymbol{r_\beta}\, {}^0\mathbf{R}_3^\circ\, {}^3\mathbf{R}_4^\circ \\ \Rightarrow {}^0\mathbf{R}_3 &= \boldsymbol{r_\beta}\, {}^0\mathbf{R}_3^\circ. \end{aligned} \tag{4.10}$$

Knowing this, Eq. (4.5) can be rewritten as :

$$\begin{aligned} {}^0\mathbf{p}_7 &= {}^0\mathbf{x}_{bs} + \boldsymbol{r_\beta}\, {}^0\mathbf{R}_3^\circ \Big({}^3\mathbf{x}_{se} + {}^3\mathbf{R}_4({}^4\mathbf{x}_{ew} + {}^4\mathbf{R}_7\, {}^7\mathbf{x}_{wt}) \Big) \\ {}^0\mathbf{R}_7 &= \boldsymbol{r_\beta}\, {}^0\mathbf{R}_3^\circ\, {}^3\mathbf{R}_4\, {}^4\mathbf{R}_7. \end{aligned} \tag{4.11}$$

Derivation of q_4 and ${}^0\mathbf{R}_3^\circ$: The last step before computing all the actual joints position values is to determine the value of q_4 and ${}^0\mathbf{R}_3^\circ$. The value of vector $\overrightarrow{\mathrm{SW}}^d \equiv {}^0\mathbf{x}_{sw}^d$ that complies with the desired task[7] is first computed (*note* : all the terms on the right of the expression are known given a task):

$$\begin{aligned} \overrightarrow{\mathrm{SW}} &= \overrightarrow{\mathrm{SB}} + \overrightarrow{\mathrm{BT}} + \overrightarrow{\mathrm{TW}} \\ \Leftrightarrow {}^0\mathbf{x}_{sw}^d &= -\,{}^0\mathbf{x}_{bs} + {}^0\mathbf{x}_7^d - {}^0\mathbf{R}_7^d\, {}^7\mathbf{x}_{wt} \end{aligned} \tag{4.12}$$

The first joint position that is computed is the one of joint 4, q_4. The law of cosines is used within triangle SEW (visible on Fig. 1.3.5).

$$\begin{aligned} \cos(q4) &= \frac{\| {}^0x_{sw}^d\|^2 - {l_{se}}^2 - {l_{ew}}^2}{2 l_{se} l_{ew}} \\ q4 &= \arccos \frac{\| {}^0x_{sw}^d\|^2 - {l_{se}}^2 - {l_{ew}}^2}{2 l_{se} l_{ew}} \end{aligned} \tag{4.13}$$

[7] Right-hand-side superscript notation $(.)^d$ stands for the value of "(.)" that complies with the desired task.

${}^0\mathbf{x}_{sw}$ can also be derived so as to make ${}^0\mathbf{R}_3^\circ$ appear:

$$\begin{aligned}
&\overrightarrow{\mathrm{SW}} = \overrightarrow{\mathrm{SE}} + \overrightarrow{\mathrm{EW}} \\
\Leftrightarrow\ & {}^0\mathbf{x}_{sw} = {}^0\mathbf{R}_3^\circ({}^3\mathbf{x}_{se} + {}^3\mathbf{R}_4\,{}^3\mathbf{x}_{ew}) \\
\Leftrightarrow\ & {}^0\mathbf{x}_{sw} = {}^0\mathbf{R}_1^\circ\,{}^1\mathbf{R}_2^\circ\,{}^2\mathbf{R}_3^\circ({}^3\mathbf{x}_{se} + {}^3\mathbf{R}_4\,{}^4\mathbf{x}_{ew}).
\end{aligned} \tag{4.14}$$

The DH parameterisation of the robot (see Section 1.3.2.1) gives us the formulas to compute rotation matrices[8] ${}^0\mathbf{R}_1^\circ$, ${}^1\mathbf{R}_2^\circ$, ${}^2\mathbf{R}_3^\circ$ and ${}^3\mathbf{R}_4$:

$${}^0\mathbf{R}_1^\circ = \begin{bmatrix} c_1^\circ & -s_1^\circ & 0 \\ s_1^\circ & c_1^\circ & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4.15}$$

$${}^1\mathbf{R}_2^\circ = \begin{bmatrix} c_2^\circ & -s_2^\circ & 0 \\ 0 & 0 & 1 \\ -s_2^\circ & -c_2^\circ & 0 \end{bmatrix} \tag{4.16}$$

$${}^2\mathbf{R}_3^\circ = \begin{bmatrix} \cos(0) & -\sin(0) & 0 \\ 0 & 0 & -1 \\ \sin(0) & \cos(0) & 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix} \tag{4.17}$$

$${}^3\mathbf{R}_4 = \begin{bmatrix} c_4 & -s_4 & 0 \\ 0 & 0 & -1 \\ s_4 & c_4 & 0 \end{bmatrix} \text{ (with } q_4 \text{ taken from Eq. (4.13))} \tag{4.18}$$

In Eq. (4.14), let us simplify the notation of the right hand side term by noting ${}^3\mathbf{x}_{se} + {}^3\mathbf{R}_4\,{}^4\mathbf{x}_{ew} = {}^3\mathbf{x}_{sw}$ and by adopting matlab matrix notation (*e.g.* 3 lines and 1 column matrix ${}^3\mathbf{x}_{sw}$ becomes:

$${}^3\mathbf{x}_{sw} = \begin{bmatrix} {}^3\mathbf{x}_{sw}(1,1) \\ {}^3\mathbf{x}_{sw}(2,1) \\ {}^3\mathbf{x}_{sw}(3,1) \end{bmatrix}).$$

[8] For the sake of brevity and concision, $\cos(q_i^\circ)$ and $\sin(q_i^\circ)$ will be noted c_i° and s_i°

In that case, the desired shoulder wrist vector computed in Eq. (4.12) can be equated to ${}^0\mathbf{R}_1^\circ\,{}^1\mathbf{R}_2^\circ\,{}^2\mathbf{R}_3^\circ\,{}^3\mathbf{x}_{sw}$, leading to a set of equations that allow to compute[9] q_1° and q_2° :

$$
\begin{aligned}
&{}^0\mathbf{x}_{sw}^d = {}^0\mathbf{R}_1^\circ\,{}^1\mathbf{R}_2^\circ\,{}^2\mathbf{R}_3^\circ\,{}^3\mathbf{x}_{sw} \\
\Rightarrow &\begin{cases}
{}^0\mathbf{x}_{sw}^d(1,1) &= {}^3\mathbf{x}_{sw}(1,1)c_1^\circ c_2^\circ - {}^3\mathbf{x}_{sw}(2,1)s_1^\circ + {}^3\mathbf{x}_{sw}(3,1)c_1^\circ s_2^\circ \\
{}^0\mathbf{x}_{sw}^d(2,1) &= {}^3\mathbf{x}_{sw}(1,1)s_1^\circ c_2^\circ + {}^3\mathbf{x}_{sw}(2,1)c_1^\circ + {}^3\mathbf{x}_{sw}(3,1)s_1^\circ s_2^\circ \\
{}^0\mathbf{x}_{sw}^d(3,1) &= -{}^3\mathbf{x}_{sw}(1,1)s_2^\circ + {}^3\mathbf{x}_{sw}(3,1)c_2^\circ
\end{cases}
\end{aligned}
\tag{4.19}
$$

Parameterised computation of q_1, q_2 and q_3 : The computation of the three first joint angles takes roots in Eq. (4.10). The term ${}^0\mathbf{R}_3$ is once again expended to account for the joint angle values :

$$
{}^0\mathbf{R}_3 = {}^0\mathbf{R}_1\,{}^1\mathbf{R}_2\,{}^2\mathbf{R}_3 = \begin{bmatrix}
c_1c_2c_3 - s_1s_3 & -c_3s_1 - c_1c_2s_3 & c_1s_2 \\
c_1s_3 + c_2c_3s_1 & c_1c_3 - c_2s_1s_3 & s_1s_2 \\
-c_3s_2 & s_2s_3 & c_2
\end{bmatrix}
$$

The right hand side of Eq. (4.10) can also be expanded in :

$$
\begin{aligned}
\boldsymbol{r_\beta}\,{}^0\mathbf{R}_3^\circ &= \left(\sin(\beta)[{}^0\mathbf{u}_{sw_\times}] - \cos(\beta)[{}^0\mathbf{u}_{sw_\times}]^2 + (\boldsymbol{I_3} + [{}^0\mathbf{u}_{sw_\times}]^2)\right){}^0\mathbf{R}_3^\circ \\
&= \sin(\beta)[{}^0\mathbf{u}_{sw_\times}]\,{}^0\mathbf{R}_3^\circ - \cos(\beta)[{}^0\mathbf{u}_{sw_\times}]^2\,{}^0\mathbf{R}_3^\circ + (\boldsymbol{I_3} + [{}^0\mathbf{u}_{sw_\times}]^2)\,{}^0\mathbf{R}_3^\circ \\
&= \sin(\beta)\boldsymbol{A_S} + \cos(\beta)\boldsymbol{B_S} + \boldsymbol{C_S},
\end{aligned}
\tag{4.20}
$$

where

$$
\begin{aligned}
\boldsymbol{A_S} &= (a_{s_{i,j}})_{(i,j)\in[\![1..3]\!]} = [{}^0\mathbf{u}_{sw_\times}]\,{}^0\mathbf{R}_3^\circ \\
\boldsymbol{B_S} &= (b_{s_{i,j}})_{(i,j)\in[\![1..3]\!]} = -[{}^0\mathbf{u}_{sw_\times}]^2\,{}^0\mathbf{R}_3^\circ \\
\boldsymbol{C_S} &= (c_{s_{i,j}})_{(i,j)\in[\![1..3]\!]} = (\boldsymbol{I_3} + [{}^0\mathbf{u}_{sw_\times}]^2)\,{}^0\mathbf{R}_3^\circ.
\end{aligned}
$$

[9] The resolution of this non linear system is not detailed here for the sake of brevity. In one of [165] appendices can be found a very concise method for this kind of system of equations, in order to consistently find the values of q_1° and q_2°.

Eq. (4.10) gives us ${}^0\mathbf{R}_3 = \boldsymbol{r_\beta}\,{}^0\mathbf{R}_3^\circ$. Therefore,

$$
\begin{aligned}
&\text{if } s_2 = \sin(q_2) \neq 0 \Leftrightarrow |c_2| = |\cos(q_2)| = |\,{}^0\mathbf{R}_3(3,3)| \neq 1,\\
&\quad q_1 = \operatorname{atan2}({}^0\mathbf{R}_3(2,3), {}^0\mathbf{R}_3(1,3))\\
&\qquad = \operatorname{atan2}(a_{s_{2,3}} s_\beta + b_{s_{2,3}} c_\beta + c_{s_{2,3}}, a_{s_{1,3}} s_\beta + b_{s_{1,3}} c_\beta + c_{s_{1,3}}),\\
&\quad q_3 = \operatorname{atan2}({}^0\mathbf{R}_3(3,2), -\,{}^0\mathbf{R}_3(3,1)),\\
&\qquad = \operatorname{atan2}(a_{s_{3,2}} s_\beta + b_{s_{3,2}} c_\beta + c_{s_{3,2}}, -a_{s_{3,1}} s_\beta - b_{s_{3,1}} c_\beta - c_{s_{3,1}}),\\
&\quad \text{if } \sin(q_1) \neq 0\\
&\qquad q_2 = \operatorname{atan2}({}^0\mathbf{R}_3(2,3)/\sin(q_1), {}^0\mathbf{R}_3(3,3))\\
&\qquad\quad = \operatorname{atan2}(\frac{1}{s_1}(a_{s_{2,3}} s_\beta + b_{s_{2,3}} c_\beta + c_{s_{2,3}}), a_{s_{3,3}} s_\beta + b_{s_{3,3}} c_\beta + c_{s_{3,3}})\\
&\quad \text{else } (\Leftrightarrow \sin(q_1) = 0)\\
&\qquad q_2 = \operatorname{atan2}({}^0\mathbf{R}_3(1,3)/\cos(q_1), {}^0\mathbf{R}_3(3,3))\\
&\qquad\quad = \operatorname{atan2}(\frac{1}{c_1}(a_{s_{1,3}} s_\beta + b_{s_{1,3}} c_\beta + c_{s_{1,3}}), a_{s_{3,3}} s_\beta + b_{s_{3,3}} c_\beta + c_{s_{3,3}})\\
&\text{else } (\Leftrightarrow \sin(q_2) = 0)\\
&\quad q_1 = 0 \text{ (arbitrarily, as joint 1 and 3 play the same role)}\\
&\quad q_3 = \operatorname{atan2}({}^0\mathbf{R}_3(2,1), {}^0\mathbf{R}_3(2,2))\\
&\qquad = \operatorname{atan2}(a_{s_{2,1}} s_\beta + b_{s_{2,1}} c_\beta + c_{s_{2,1}}, a_{s_{2,2}} s_\beta + b_{s_{2,2}} c_\beta + c_{s_{2,2}})\\
&\quad q_2 = \operatorname{atan2}(0, {}^0\mathbf{R}_3(3,3))\\
&\qquad = \operatorname{atan2}(0, a_{s_{3,3}} s_\beta + b_{s_{3,3}} c_\beta + c_{s_{3,3}})
\end{aligned}
\tag{4.21}
$$

Parameterised computation of q_5, q_6 and q_7 : With the values of the four first joint positions, the value of ${}^4\mathbf{R}_7^d$ can be computed by composing ${}^4\mathbf{R}_0 = {}^0\mathbf{R}_4{}^\intercal = ({}^0\mathbf{R}_3\,{}^3\mathbf{R}_4)^\intercal = (\boldsymbol{r_\beta}\,{}^0\mathbf{R}_3^\circ\,{}^3\mathbf{R}_4)^\intercal$ with ${}^0\mathbf{R}_7^d$, as such:

$$
{}^4\mathbf{R}_7^d = (\boldsymbol{r_\beta}\,{}^0\mathbf{R}_3^\circ\,{}^3\mathbf{R}_4)^\intercal\,{}^0\mathbf{R}_7^d = {}^3\mathbf{R}_4{}^\intercal(\boldsymbol{r_\beta}\,{}^0\mathbf{R}_3^\circ)^\intercal\,{}^0\mathbf{R}_7^d \tag{4.22}
$$

Similarly as before, let us write things so as to make β appear more clearly. Using Eq. (4.20),

$$\begin{aligned} {}^3\mathbf{R}_4{}^\intercal({\boldsymbol{r}_{\boldsymbol{\beta}}}\,{}^0\mathbf{R}_3^\circ)^\intercal\,{}^0\mathbf{R}_7^d &= {}^3\mathbf{R}_4{}^\intercal(\sin(\beta)\boldsymbol{A_S} + \cos(\beta)\boldsymbol{B_S} + \boldsymbol{C_S})^\intercal\,{}^0\mathbf{R}_7^d \\ &= \sin(\beta)\boldsymbol{A_W} + \cos(\beta)\boldsymbol{B_W} + \boldsymbol{C_W}. \end{aligned}$$

With

$$\begin{aligned} \boldsymbol{A_W} = (a_{w_{i,j}})_{(i,j)\in[\![1..3]\!]} &= {}^3\mathbf{R}_4{}^\intercal\boldsymbol{A_S^\intercal}\,{}^0\mathbf{R}_7^d \\ \boldsymbol{B_W} = b_{w_{i,j}})_{(i,j)\in[\![1..3]\!]} &= {}^3\mathbf{R}_4{}^\intercal\boldsymbol{B_S^\intercal}\,{}^0\mathbf{R}_7^d \\ \boldsymbol{C_W} = (c_{w_{i,j}})_{(i,j)\in[\![1..3]\!]} &= {}^3\mathbf{R}_4{}^\intercal\boldsymbol{C_S^\intercal}\,{}^0\mathbf{R}_7^d. \end{aligned}$$

From the DH model of the system, matrix ${}^4\mathbf{R}_7$ can be expended to account for the joint angle values q_5, q_6 and q_7:

$${}^4\mathbf{R}_7 = {}^4\mathbf{R}_5\,{}^5\mathbf{R}_6\,{}^6\mathbf{R}_7 = \begin{bmatrix} c_5c_6c_7 - s_5s_7 & -c_7s_5 - c_5c_6s_7 & c_5s_6 \\ -c_7s_6 & s_6s_7 & c_6 \\ -c_5s_7 - c_6c_7s_5 & c_6s_5s_7 - c_5c_7 & -s_5s_6 \end{bmatrix}$$

Therefore,

$$
\begin{aligned}
&\text{if } s_6 = \sin(q_6) \neq 0 \Leftrightarrow |c_6| = |\cos(q_6)| = |\,{}^4\mathbf{R}_7(2,3)| \neq 1,\\
&\quad q_5 = \operatorname{atan2}(-\,{}^4\mathbf{R}_7(3,3), {}^4\mathbf{R}_7(1,3))\\
&\qquad = \operatorname{atan2}(-a_{w_{3,3}} s_\beta - b_{w_{3,3}} c_\beta - c_{w_{3,3}}, a_{w_{1,3}} s_\beta + b_{w_{1,3}} c_\beta + c_{w_{1,3}}),\\
&\quad q_7 = \operatorname{atan2}({}^4\mathbf{R}_7(2,2), -\,{}^4\mathbf{R}_7(2,1)),\\
&\qquad = \operatorname{atan2}(a_{w_{2,2}} s_\beta + b_{w_{2,2}} c_\beta + c_{w_{2,2}}, -a_{w_{2,1}} s_\beta - b_{w_{2,1}} c_\beta - c_{w_{2,1}}),\\
&\quad \text{if } \sin(q_5) \neq 0\\
&\qquad q_6 = \operatorname{atan2}(-\,{}^4\mathbf{R}_7(3,3)/\sin(q_5), {}^4\mathbf{R}_7(2,3))\\
&\qquad\quad = \operatorname{atan2}(\frac{-1}{s_5}(a_{w_{3,3}} s_\beta + b_{w_{3,3}} c_\beta + c_{w_{3,3}}), a_{w_{2,3}} s_\beta + b_{w_{2,3}} c_\beta + c_{w_{2,3}})\\
&\quad \text{else } (\Leftrightarrow \sin(q_5) = 0)\\
&\qquad q_6 = \operatorname{atan2}({}^4\mathbf{R}_7(1,3)/\cos(q_5), {}^4\mathbf{R}_7(2,3))\\
&\qquad\quad = \operatorname{atan2}(\frac{1}{c_5}(a_{w_{1,3}} s_\beta + b_{w_{1,3}} c_\beta + c_{w_{1,3}}), a_{w_{2,3}} s_\beta + b_{w_{2,3}} c_\beta + c_{w_{2,3}})\\
&\text{else } (\Leftrightarrow \sin(q_6) = 0)\\
&\quad q_5 = 0 \text{ (arbitrarily, as joint 5 and 7 play the same role)}\\
&\quad q_7 = \operatorname{atan2}({}^4\mathbf{R}_7(3,1), {}^4\mathbf{R}_7(3,2))\\
&\qquad = \operatorname{atan2}(a_{w_{3,1}} s_\beta + b_{w_{3,1}} c_\beta + c_{w_{3,1}}, a_{w_{3,2}} s_\beta + b_{w_{3,2}} c_\beta + c_{w_{3,2}})\\
&\quad q_6 = \operatorname{atan2}(0, {}^4\mathbf{R}_7(2,3))\\
&\qquad = \operatorname{atan2}(0, a_{w_{2,3}} s_\beta + b_{w_{2,3}} c_\beta + c_{w_{2,3}})
\end{aligned}
\tag{4.23}
$$

The demonstration above showed how to compute one solution of joint angles that comply with the task of positioning and orientating the robot tip reference frame in a desired pose while using a given elbow angle β. However, due to the trigonometric nature of the functions of the forward geometric model, there exist multiple solutions. The LBR iiwa, with its anthropomorphic structure, actually possesses 8 different

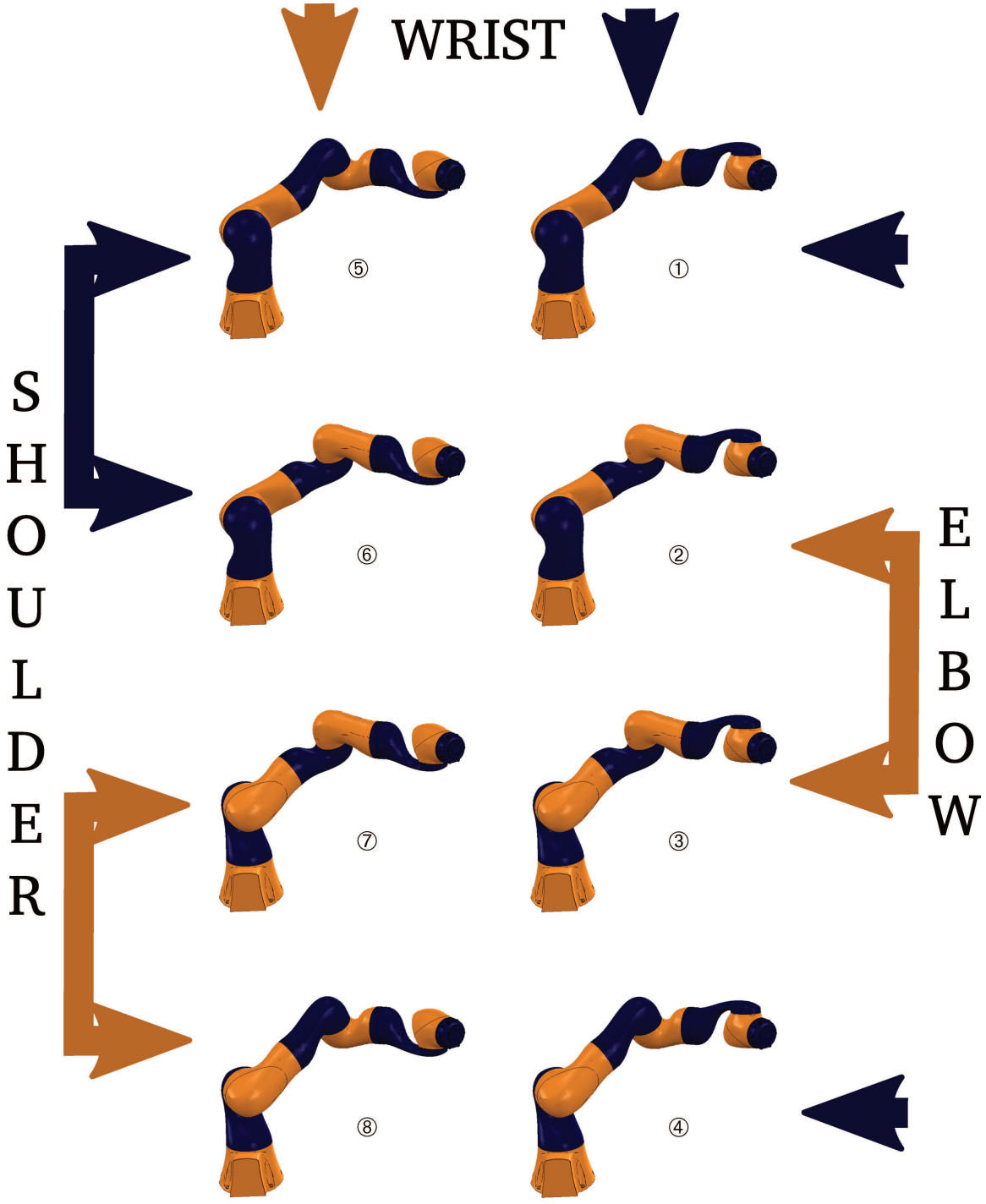


Figure 4.6.3: Eight different articular configurations complying with the same task and elbow angle

①	②	③	④	⑤	⑥	⑦	⑧
q_1	q_1	$q_1 + \pi$	$q_1 + \pi$	q_1	q_1	$q_1 + \pi$	$q_1 + \pi$
q_2	q_2	$-q_2$	$-q_2$	q_2	q_2	$-q_2$	$-q_2$
q_3	$q_3 + \pi$	q_3	$q_3 + \pi$	q_3	$q_3 + \pi$	q_3	$q_3 + \pi$
q_4	$-q_4$	$-q_4$	q_4	q_4	$-q_4$	$-q_4$	q_4
q_5	$q_5 + \pi$	$q_5 + \pi$	q_5	$q_5 + \pi$	q_5	q_5	$q_5 + \pi$
q_6	q_6	q_6	q_6	$-q_6$	$-q_6$	$-q_6$	$-q_6$
q_7	q_7	q_7	q_7	$q_7 + \pi$	$q_7 + \pi$	$q_7 + \pi$	$q_7 + \pi$

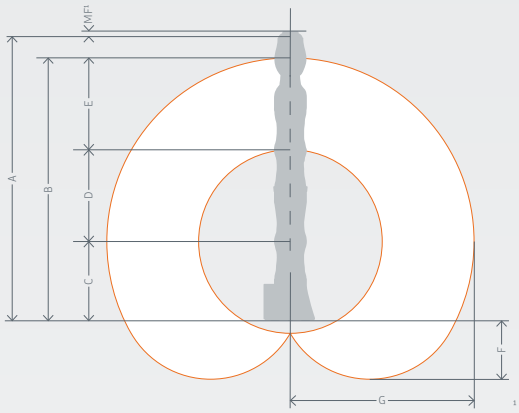
Table 4.6.1: The 8 possible configurations seen on Fig. 4.6.3

solutions to this problem, for the same elbow angle value, as can be seen on Fig. 4.6.3. To compute the other possibilities, one can use Table 4.6.1.

Appendix 3 : LBR iiwa specifications

The following document in part of a commercial brochure of the LRB iiwa robot.

Technical data



[1] dependent on the media flange option

Workspace	Dimensions A	Dimensions B	Dimensions C	Dimensions D	Dimensions E	Dimensions F	Dimensions G	Volume
LBR iiwa 7 R800	1,266 mm	1,140 mm	340 mm	400 mm	400 mm	260 mm	800 mm	1.7 m^3
LBR iiwa 14 R820	1,306 mm	1,180 mm	360 mm	420 mm	400 mm	255 mm	820 mm	1.8 m^3

LBR iiwa	LBR iiwa 7 R800	LBR iiwa 14 R820
Rated payload	7 kg	14 kg
Number of axes	7	7
Wrist variant	In-line wrist	In-line wrist
Mounting flange A7	DIN ISO 9409-1-A50	DIN ISO 9409-1-A50
Installation position	any	any
Positioning accuracy (ISO 9283)	± 0.1 mm	± 0.1 mm
Axis-specific torque accuracy	± 2 %	± 2 %
Weight	23.9 kg	29.9 kg
Protection rating	IP 54	IP 54

Axis data / Range of motion		**LBR iiwa 7 kg** Maximum torque	**LBR iiwa 7 kg** Maximum velocity	Maximum torque	**LBR iiwa 14 kg** Maximum velocity
Axis 1 (A1)	± 170°	176 Nm	98°/s	320 Nm	85°/s
Axis 2 (A2)	± 120°	176 Nm	98°/s	320 Nm	85°/s
Axis 3 (A3)	± 170°	110 Nm	100°/s	176 Nm	100°/s
Axis 4 (A4)	± 120°	110 Nm	130°/s	176 Nm	75°/s
Axis 5 (A5)	± 170°	110 Nm	140°/s	110 Nm	130°/s
Axis 6 (A6)	± 120°	40 Nm	180°/s	40 Nm	135°/s
Axis 7 (A7)	± 175°	40 Nm	180°/s	40 Nm	135°/s

Programmable Cartesian stiffness	
Min. (X, Y, Z)	0.0 N/m
Max. (X, Y, Z)	5,000 N/m
Min. (A, B, C)	0.0 N/rad
Max. (A, B, C)	300 Nm/rad

KUKA Sunrise Cabinet	
Processor	Quad-core processor
Hard drive	SSD
Interfaces	USB, EtherNet, DVI-I
Protection rating	IP20
Dimensions (D x W x H)	500 mm x 483 mm x 190 mm
Weight	23 kg

Power supply connection	
Rated supply voltage	AC 110 V to 230 V
Permissible tolerance of rated voltage	± 10 %
Mains frequency	50 Hz ± 1 Hz or 60 Hz ± 1 Hz
Mains-side fusing	2 x 16 A slow-blowing

30,000 operating hours

Appendix 4 : Résumé en Français

4.7 Introduction

Tout comme dans d'autres secteurs, l'industrie de production aéronautique a connu des changements majeurs avec l'automatisation de ses procédés. Bien que de nombreuses opérations soient encore effectuées par des opérateurs humains, de plus en plus de machines colonisent les zones de production pour la réalisation des tâches les plus pénibles et répétitives. Un avantage notoire de ces machines est leur grande répétabilité qui assure une bonne qualité de réalisation des opérations. D'abord occupée par d'encombrantes machines spéciales, l'industrie de production automatisée a tendance à s'orienter vers des solutions plus versatiles. Les parties opérationnelles de ces machines spécialisées ont été transférées dans des effecteurs, eux-mêmes montés sur des robots sériels articulés. Des robots gros porteurs ont naturellement été utilisés pour assurer cette transition, ce qui a permis de fortement diminuer les coûts de mise en service, de par leur généricité. Aujourd'hui, la tendance de flexibilisation et de miniaturisation des solutions productiques amène à repenser les espaces de production pour permettre la cohabitation sécuritaire des machines et des hommes. L'arrivée de robots plus petits, mobiles et sécuritaires sur la scène de la production favorise cette tendance. Les robots collaboratifs, dont la puissance est limitée, sont dotés de capteurs leur permettant de détecter la présence d'obstacles, afin de garantir la sécurité des humains se trouvant aux alentours.

On s'intéresse dans cette thèse à l'utilisation de systèmes redondants, collaboratifs et mobiles (bras articulés montés sur plateformes mobiles), dans un environnement de production aéronautique peuplé d'humains, pour la réalisation d'opérations

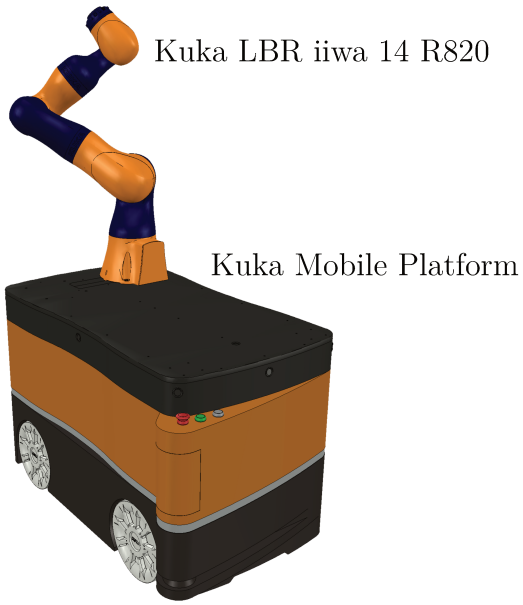


Figure 4.7.1: LBR iiwa 14 R820 monté sur une plateforme mobile Kuka (KMP)

d'assemblage.

Du point de vue des process, l'utilisation de ces systèmes, souvent beaucoup moins imposants et rigides que leurs homologues non collaboratifs, est jalonnée de défis. La grande souplesse mécanique et les faibles couples qui les caractérisent peuvent induire des imprécisions de positionnement et une incapacité à soutenir l'intensité d'une interaction physique. Ce contexte induit également un besoin d'autonomie de ces systèmes, qui sont amenés à travailler dans des environnements en perpétuelle évolution.

Une analogie peut être utilisée pour décrire simplement la manière dont des procédés industriels peuvent être réalisés par des systèmes autonomes dans un environnement encombré et dynamique. La Fig. 4.7.2 illustre le raisonnement et l'organisation des tâches qu'un humain peut avoir pour planter un clou dans un mur.

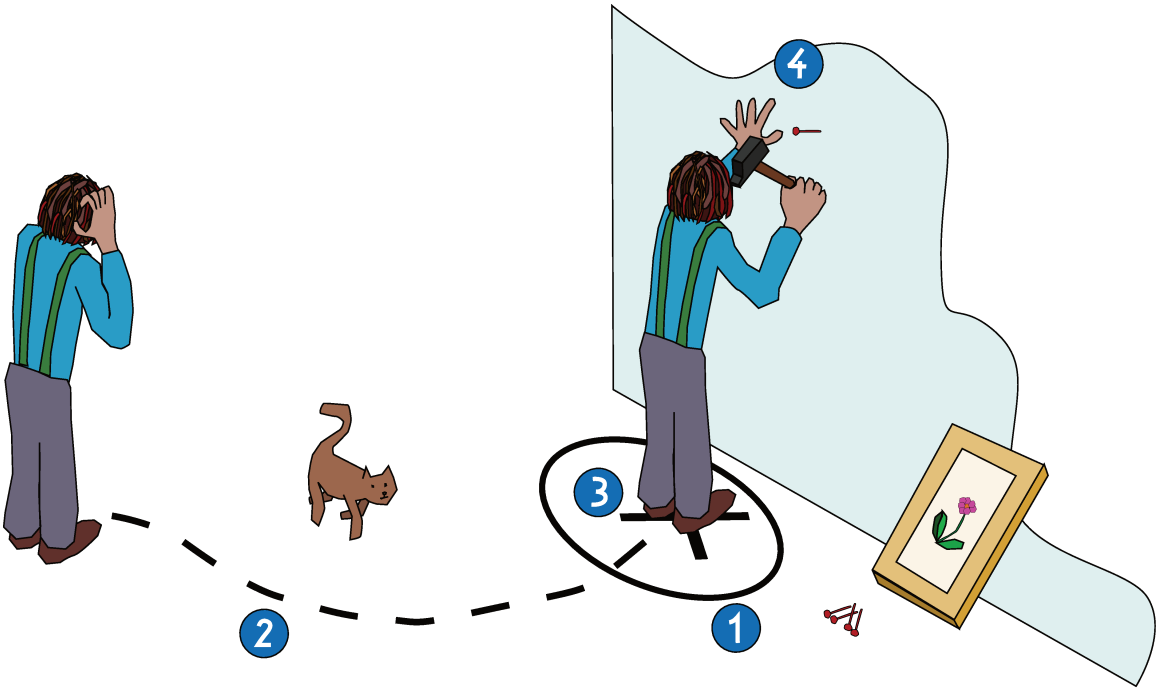


Figure 4.7.2: Une analogie simple permettant d'illustrer les étapes qu'un système mobile et redondant peut suivre pour remplir une tâche industrielle.

1. A partir de l'objectif de planter un clou à un certain emplacement, l'humain détermine à peu près l'endroit où il positionnera ses pieds pour pouvoir atteindre la position destinée au clou. De son côté, un système autonome et mobile doit déterminer une région de positionnement de sa base lui permettant l'atteignabilité par son effecteur de la définition géométrique de la tâche qui lui est assignée.

2. L'humain doit ensuite se déplacer jusqu'à la région identifiée tout en évitant les collisions avec les meubles, les murs, ou son animal de compagnie. Le système industriel doit ensuite se déplacer de manière sécuritaire jusqu'à sa destination. L'environnement peut être peuplé d'obstacles statiques ou dynamiques.

3. Après avoir grossièrement rejoint sa destination, l'humain recherche une posture qui lui permet de planter le clou dans de bonnes conditions. En général, cette

étape est empirique ou provient d'un apprentissage. L'humain procède par essai-erreur, en corrigeant sa posture jusqu'à en atteindre une qui lui semble satisfaisante du point de vue de l'efficacité, de la précision, ou de sa capacité à taper suffisamment fort avec son marteau. Par exemple, il est peu probable qu'il choisisse de planter le clou en tournant le dos au mur. Notre système autonome est redondant, et pourrait positionner correctement son effecteur avec une grande variété de postures. Une de ces postures doit être choisie pour lui permettre la réalisation de sa tâche dans de bonnes conditions (en pratique, peut être que ce choix peut être réalisé avant pour le système robotique).

4. Après avoir choisi une posture convenable, l'humain peut planter le clou dans le mur. Le système industriel peut commencer à réaliser le procédé lorsqu'il a atteint une posture convenable.

Dans cette thèse, une formulation de la redondance cinématique est d'abord présentée. Le formalisme associé permet de simplifier l'exploitation de la liberté que ces systèmes possèdent sur le choix des postures à utiliser pour réaliser des tâches de placement statique de l'effecteur. Ce formalisme est ensuite exploité pour améliorer et caractériser le comportement en déformation et la capacité d'application d'efforts des systèmes redondants sériels. Enfin, le sujet de la planification des mouvements de systèmes robotisés dans un environnement dynamique et encombré est considéré. La solution présentée adapte l'algorithme bien connu des *Probabilistic RoadMaps* pour y inclure une anticipation des trajectoires des obstacles dynamiques. Cette solution permet de planifier des mouvements sécuritaires, peu intrusifs et efficaces jusqu'à la destination.

4.8 Une formulation de la redondance de positionnement

4.8.1 Espaces de redondance

Contexte : Dans le secteur de production aéronautique, les robots sont utilisés pour réaliser des opérations à haute valeur ajoutée, nécessitant une répétabilité et

une précision importante. Les tolérances mécaniques typiquement demandée pour ces opérations sont en dessous de 0.2 mm. Pourtant, les performances typiques des robots industriels sont de l'ordre de 0.3-0.5 mm en termes de répétabilité et de 1-3 mm en termes de précision. Par conséquent, ces opérations peuvent être réalisées après que le robot se soit positionné précisément de manière statique, en utilisant des capteurs externes (laser tracker, caméra, etc...). Dans le secteur de production aéronautique comme dans d'autres secteurs, de nombreux procédés robotisés peuvent être décrits comme des tâches de positionnement statique. Les opérations de pick and place définissent un placement de la pince pour attraper et un placement pour relâcher un objet. Les opérations d'assemblage telles que le rivetage, le vissage, ou même les opérations de perçage font partie de ce type de tâche de positionnement statique de l'effecteur. Après s'être positionné précisément, on demande généralement au robot de rester immobile pendant que son outil réalise le procédé (qu'il s'agisse de perçage, rivetage, vissage, ou autre). Pendant l'interaction, la tâche réalisé par le robot est en apparence la plus simple: maintenir une configuration fixe jusqu'à ce que le procédé soit terminé. Dans ce contexte, le système peut naturellement être appelé un positionneur. Le but pour le bras robotique est de fournir un moment non perturbé, laissant le temps à l'outil de réaliser son procédé dans les meilleures conditions. La moindre déviation géométrique à l'interface du procédé peut ruiner la qualité du résultat, ou engendrer de la casse.

Afin de réaliser ces tâches de positionnement, notre cas d'étude implique l'utilisation d'un système composé d'une plate-forme mobile (KMP) et d'un robot sériel à 7 degrés de liberté (LBR-iiwa). Ensemble, ces deux sous-systèmes accumulent 10 degrés de liberté, ce qui rend le système complet extrêmement redondant par rapport à tous types de tâches de positionnement statique de l'effecteur.

Objectifs: La redondance est parfois perçue comme une source supplémentaire de complexité. Cependant, la redondance est aussi une source de possibilité et de flexibilité. Elle fournit un choix dans la configuration articulaire à utiliser. Plus le degré de redondance est important, plus il y a de possibilités. Par conséquent, la présence de la redondance implique de devoir prendre des décisions, qui n'existaient pas pour

des robots non redondant. Nous voulons rendre les possibilités que la redondance offre clairement identifiables pour les utilisateurs de ces robots. Nous avons identifié un besoin évident d'une formulation simple de la redondance au niveau du positionnement, qui permette de connaître l'étendue des solutions que la redondance offre. De plus, cette formulation devra permettre un calcul rapide et sans approximation de la configuration articulaire associée.

Expression de la redondance en positionnement par des outils différentiels: Traditionnellement, la résolution de redondance est réalisée en considérant la tâche comme étant une consigne de vitesse ou d'accélération Cartésienne de l'effecteur. Le Jacobien cinématique des robots en est un outil fondamental, car son inversion permet de calculer les vitesses ou les accélérations articulaires générant des déplacements de l'effecteur se conformant à la consigne. Pour déterminer les positions articulaires satisfaisant une tâche de positionnement, ces méthodes ont recours à une phase d'intégration temporelle de la consigne articulaire. Si une tâche $\mathbf{t}$ consiste à maintenir l'effecteur statique ($\dot{\mathbf{t}} = \mathbf{0}$), un manipulateur redondant peut théoriquement déplacer ses articulations sans perturber la tâche. Les vitesses articulaires que ce manipulateur peut utiliser sans perturber la tâche sont à sélectionner dans le noyau du Jacobien associé à la tâche. Pour $\dot{\mathbf{q}}_0 \in \mathbb{R}^N$:

$$\dot{\mathbf{q}} = \boldsymbol{J_t}^{\dagger}\dot{\mathbf{t}} + (\boldsymbol{I} - \boldsymbol{J_t}^{\dagger}\boldsymbol{J_t})\dot{\mathbf{q}}_0 \tag{4.24}$$

$$= (\boldsymbol{I} - \boldsymbol{J_t}^{\dagger}\boldsymbol{J_t})\dot{\mathbf{q}}_0, \tag{4.25}$$

correspond à une vitesse articulaire qui générera un mouvement interne du manipulateur vis à vis de la tâche. L'intégration de cette vitesse au court du temps devrait pouvoir permettre de trouver un ensemble de positions articulaires permettant de se conformer à la tâche $\mathbf{t}$. De manière équivalente, le développement limité de la fonction de cinématique directe $\mathbf{F_t}$ associée à la tâche $\mathbf{t}$ aux alentours d'une configuration articulaire $\mathbf{q}$ permet d'exprimer la variation de la tâche. La variation de la tâche $\boldsymbol{\delta}\mathbf{t}$

associée à un déplacement articulaire $\boldsymbol{\delta}\mathbf{q}$ s'exprime :

$$\boldsymbol{\delta}\mathbf{t} = \mathbf{F}_\mathbf{t}(\mathbf{q}+\boldsymbol{\delta}\mathbf{q}) - \mathbf{F}_\mathbf{t}(\mathbf{q}) = \boldsymbol{J}_\mathbf{t}\boldsymbol{\delta}\mathbf{q} + \begin{bmatrix}\mathcal{O}(\|\boldsymbol{\delta}\mathbf{q}\|^2) & \dots & \mathcal{O}(\|\boldsymbol{\delta}\mathbf{q}\|^2)\end{bmatrix}^\intercal . \tag{4.26}$$

Par conséquent, en choisissant un déplacement articulaire dans le noyau de la matrice Jacobienne de la tâche (*i.e.* $\boldsymbol{\delta}\mathbf{q} \in \ker \boldsymbol{J}_\mathbf{t}$), la variation de tâche devient :

$$\boldsymbol{\delta}\mathbf{t} = \begin{bmatrix}\mathcal{O}(\|\boldsymbol{\delta}\mathbf{q}\|^2) \\ \vdots \\ \mathcal{O}(\|\boldsymbol{\delta}\mathbf{q}\|^2)\end{bmatrix},$$

Ce qui tend vers 0 si $\boldsymbol{\delta}\mathbf{q}$ est petit.

Imprécisions due à la dérive numérique: En répétant successivement la procédure décrite précédemment, on peut trouver des configurations articulaires qui se conforment à la tâche $\mathbf{t}$ avec une imprécision de l'ordre de grandeur des termes restants du développement limité. Pour améliorer cette précision, le développement limité pourrait être réalisé au n^{ime} ordre ($n > 1$) dans la phase de cinématique inverse. Cependant, aucune procédure utilisant cette stratégie ne peut déterminer des résultats précis de manière efficace. Les solutions trouvées en répétant ces calculs souffrent d'évidente dérives numériques pouvant amener à des imprécisions faites sur la réalisation de la tâche.

Exploration incomplète l'espace de solutions: Un autre point important, mise à part l'imprécision de la solution basée sur une approche différentielle, est le fait que ces approches ne permettent pas d'explorer efficacement l'ensemble des possibilités offertes par la redondance. Dans les procédures décrites ci-avant, un moyen de choisir $\boldsymbol{\delta}\mathbf{q} \in \ker \boldsymbol{J}_\mathbf{t}$ est de générer un vecteur de vitesses articulaires $\dot{\mathbf{q}}_0 \in \mathbb{R}^N$ et de le projeter dans le noyau de la matrice Jacobienne associée à la tâche grâce au projecteur $(\boldsymbol{I} - \boldsymbol{J}_\mathbf{t}^\dagger \boldsymbol{J}_\mathbf{t})$. Cependant, en choisissant arbitrairement $\dot{\mathbf{q}}_0$, il ne sera pas aisé d'explorer l'espace des solutions articulaires, d'autant moins si la redondance du système vis à vis de la tâche est d'ordre $s > 1$. Au mieux, un espace de dimension

1 pourra être échantillonné de manière discrète et incomplète. Une autre solution, plus coûteuse, mais plus exploratoire, pourrait consister à calculer la décomposition en valeurs singulières (SVD) de la matrice Jacobienne de la tâche afin d'identifier localement, vis à vis d'une configuration donnée, toutes les directions articulaires dans lesquelles le robot peut se déplacer sans perturber la tâche (à l'imprécision des restes du développement de Taylor près). Faire des petits pas articulaires dans les directions suggérées par les vecteurs propres de la matrice Jacobienne associés à son noyau pourrait permettre d'explorer de manière plus rigoureuse cet espace de solutions. Cependant, un inconvénient commun aux stratégies de résolutions différentielles est l'absence de cyclicité[28]. Cette propriété rend un échantillonnage représentatif de l'espace de solution difficile à trouver.

Pas d'identification des frontières de l'espace de solutions: Un autre inconvénient des stratégies différentielles, qui est également lié à la cyclicité, est le fait que les frontières de l'espace de solutions ne peuvent pas facilement être exprimées ou trouvées. Ces limites sont pourtant une information primordiale pour pouvoir couvrir une partie représentative de l'espace de solutions.

Sources de redondance de positionnement: Dans notre étude, la redondance est définie pour un manipulateur et une tâche de positionnement. Elle émane donc de deux sources étroitement liées. La présence de capacités supplémentaires de mouvements (actionneurs supplémentaires) est une source, le retrait de contraintes géométriques sur la tâche en est une autre. Un manipulateur à n degrés de liberté indépendant, utilisé pour une tâche $\mathbf{t}$ de positionnement de l'effecteur, de dimension $m < n$ est qualifié de redondant d'ordre $s = n - m$. La dimension du noyau de la matrice Jacobienne associée à la tâche sera égale à s dans les configurations non singulières. Cela signifie que dans une configuration non singulière, il existe une base de s déplacements articulaires indépendants (les vecteurs de vitesses articulaires engendrant le noyau du Jacobien, et issus de sa décomposition en valeurs singulières en forment une) qui maintiendront la conformité du positionnement de l'effecteur vis à vis de $\mathbf{t}$. N'importe quelle combinaison linéaire de ces s vecteurs de déplacements articulaires n'engendrera aucun déplacements de l'effecteur. Il existe toujours une base

de vecteurs indépendants de vitesses articulaires qui est directement liée aux capacités supplémentaires de mouvement (actionneurs supplémentaires) ou aux libérations de contraintes géométriques.

Espaces de redondance: Dans cette thèse, on choisit de traiter la redondance par une approche différente. La résolution de redondance au niveau de la vitesse ou de l'accélération parait inadaptée pour la réalisation de tâches nécessitant le positionnement statique de l'effecteur. Une formulation au niveau de la position semble plus appropriée.

La formulation d'un *espace de redondance*, c'est à dire un espace paramétré par des variables indépendantes, est utilisé pour représenter l'espace des solutions disponibles au robot pour réaliser sa tâche de positionnement. Soit $\boldsymbol{\alpha} = \begin{bmatrix} \alpha_1 & ... & \alpha_s \end{bmatrix}^T$, un vecteur à s éléments représentant la position du manipulateur dans l'espace de redondance. Chaque α_i représente un paramètre de redondance. Le fait d'imposer la position du robot dans l'espace de redondance permet d'enlever toute ambiguïté dans le problème de géométrie inverse en portant le nombre d'équations à $m+s=n$ dans un système à n inconnues. Une configuration articulaire peut être rigoureusement calculée à partir du couple *position admissible dans l'espace de redondance - tâche de positionnement de l'effecteur* $(\boldsymbol{\alpha}, \mathbf{t})$. En changeant la position dans l'espace de redondance (c'est à dire la valeur des paramètres de redondance), on change la position articulaire du système tout en conservant la conformité du positionnement de l'effecteur vis à vis de la tâche. Les modèles géométriques directs des robots sont généralement non linéaires, et leur inversion analytique peut être complexe, même pour des systèmes non redondants. Cependant, pour certains types de systèmes redondants, des formulations analytiques permettent de résoudre ce problème d'inversion.

Sans être une règle absolue, la paramétrisation de l'espace de redondance peut souvent s'inspirer des contraintes géométriques libérées par rapport à la tâche absolument contrainte (imposant la position et l'orientation de l'effecteur) et/ou des capacités de mouvements internes ou externes du robot (qui proviennent plus généralement d'axes supplémentaires). Il existe souvent plus d'une formulation d'espaces de

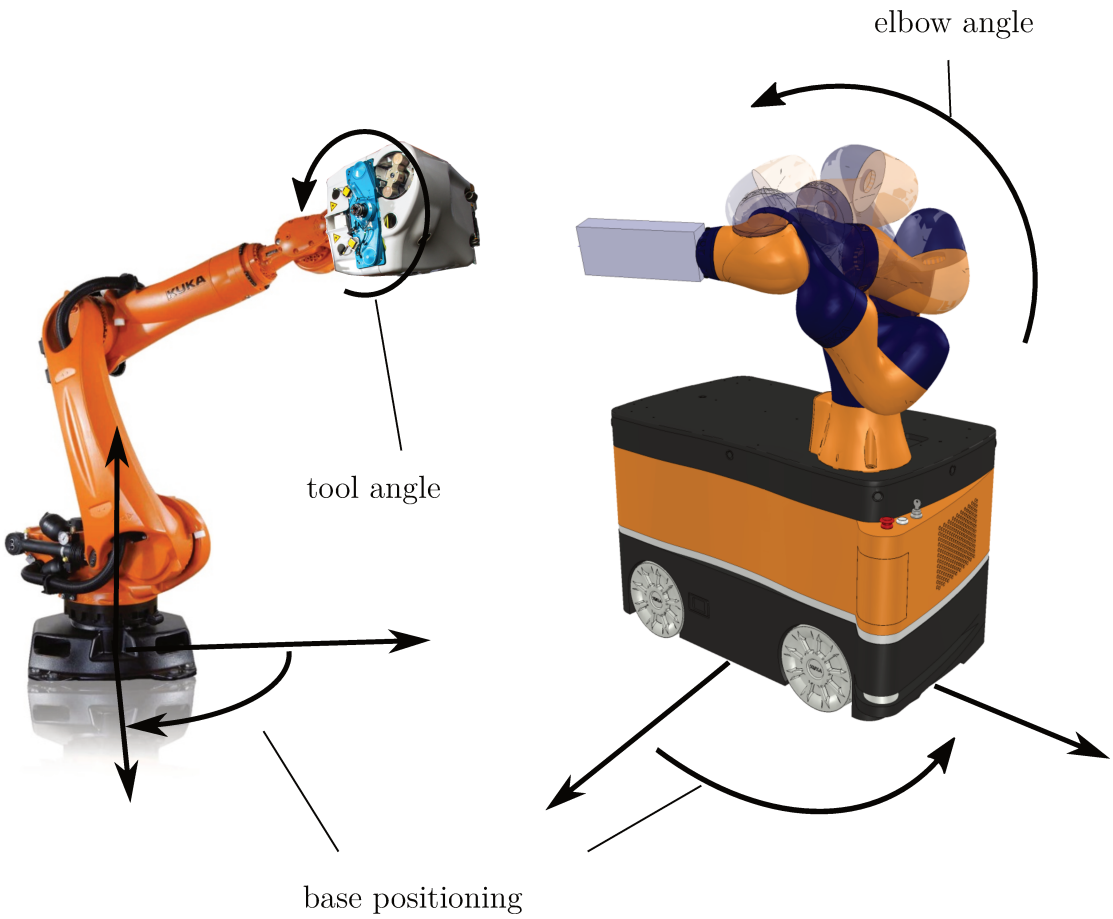


Figure 4.8.1: Exemples de paramétrisations de tâches de positionnement statique de l'effecteur pour deux systèmes différents. Sur la gauche, la tâche consiste à diriger un axe de l'effecteur (par exemple pour réaliser du perçage), ce qui libère une rotation autour de ce même axe. Sur la droite, on choisit de réaliser une tâche de positionnement contraignant complètement la position, et l'orientation de l'effecteur. Le positionnement de la base des deux robots reste libre, et son paramétrage vient augmenter la dimension de l'espace de redondance.

redondance pour un couple système-tâche donné. Une propriété nécessaire de cette paramétrisation est le fait que quel que soit le couple admissible $(\boldsymbol{\alpha}, \mathbf{t})$, il n'existe qu'un nombre fini de solutions articulaires en dehors des singularités.

Avantages d'une formulation d'espaces de redondance: Il y a plusieurs raisons d'utiliser les espaces de redondances dans ce contexte. D'une part, il est primordial d'avoir une paramétrisation succincte et non ambiguë de l'espace de solutions, qui peut être facilement comprise. Cela aide à décomplexifier la notion complexe de redondance, et facilite la communication sur le sujet. D'autre part, cette formulation permet d'exposer l'étendue de l'espace de solutions du robot, ce qui peut aider le choix d'une configuration articulaire, comme cela est fait dans les chapitres suivants. Cela permet également de vérifier rapidement l'atteignabilité du robot, ou l'existence d'une solution. De plus, lorsqu'elle est connue, le recourt à une solution analytique du problème de géométrie inverse permet de déterminer rapidement et de manière exacte les configurations se conformant à la tâche donnée.

4.8.2 Exemples de paramétrisation de la redondance appliqué à un manipulateur mobile

Pour le système qui nous est donné d'étudier et pour le type de tâches de positionnement statique souvent réalisées dans le secteur de production aéronautique, on peut identifier des paramétrages de la redondance.

Redondance de l'angle de coude : Le LBR iiwa est un bras robotisé sériel possédant 7 articulations (liaisons pivots) indépendantes. Sa structure cinématique peut être perçue comme la composition de 3 articulations : une liaison sphérique - qui englobe les 3 premiers axes du robot, qui sont concourants - est attachée à une liaison pivot - correspondant à l'axe 4 du robot - qui est elle-même attachée à une liaison sphérique - qui englobe les 3 derniers axes du robot, eux aussi concourants. Cette structure particulière le classe dans une catégorie de robots 7 axes anthropomorphiques appelée S-R-S (pour Spherical - Rotary - Spherical). Ces robots sont redondants vis à vis de la tâche consistant à positionner et orienter complètement leur segment terminal.

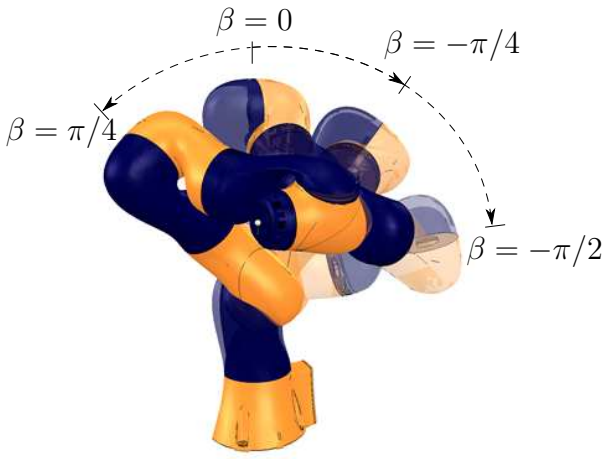


Figure 4.8.2: Paramétrisation de "l'angle de coude".

Cette tâche à 6 contraintes géométriques peut être réalisée dans un espace de solution à une dimension qui est paramétrable par "l'angle de coude" [39–41], que l'on note $\beta \in [-\pi, \pi[$ ici. La paramétrisation de cette redondance est visualisable en Fig. 1.3.6.

Orientation autour de l'axe $\mathbf{z_{tcp}}$: Une autre redondance, qui est présente pour de nombreuses tâches à symétrie de révolution (perçage, rivetage, vissage...), est celle qui libère la rotation autour de l'axe $\mathbf{z_{tcp}}$ du TCP. Elle est visible en Fig. 4.8.3. Elle est paramétrable par un angle que l'on note ici a_{tcp}.

Positionnement de la base du manipulateur : Une autre redondance qui peut être facilement identifiée sur notre système est celle du positionnement et orientation dans le plan de la base du bras manipulateur. Cette redondance est disponible lorsque le bras robotique est monté sur la base mobile. On note ici x_b, y_b et θ_b, les 3 paramètres associés à ce positionnement. Cette redondance est visualisable en Fig. 4.8.4

La combinaison de toutes ces redondances peut être synthétisée dans un espace de redondance paramétré par $\boldsymbol{\alpha} = (\beta, a_{tcp}, x_b, y_b, \theta_b)$, comme on peut l'observer en Fig. 4.8.5.

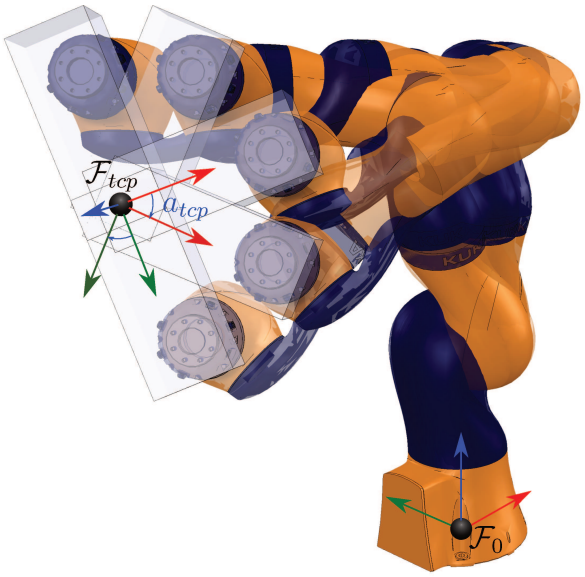


Figure 4.8.3: Redondance sur l'orientation de l'axe z_{tcp} du TCP.

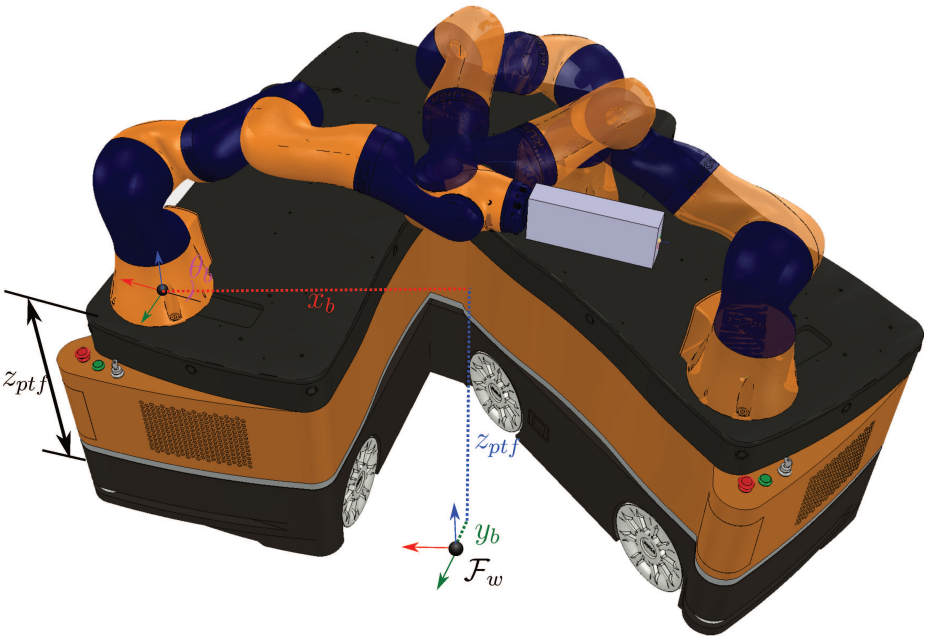


Figure 4.8.4: Redondance sur le positionnement de la plateforme mobile.

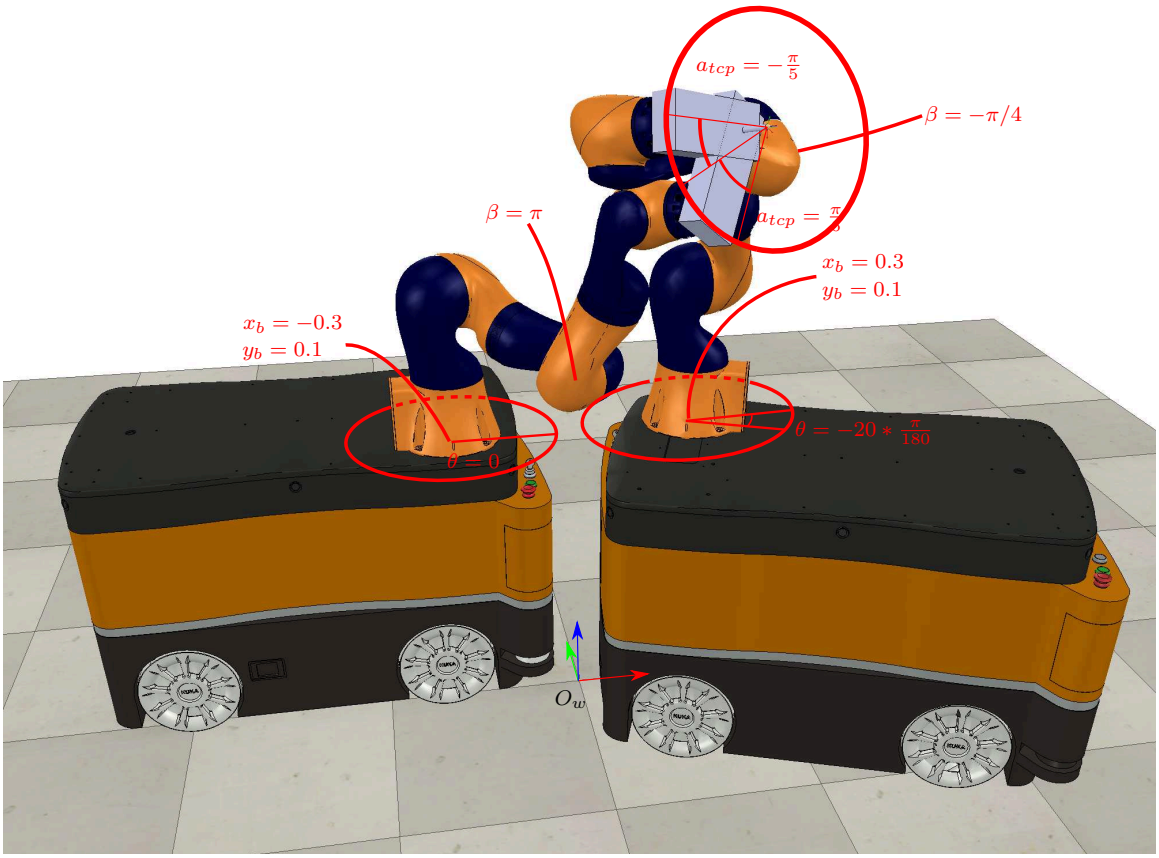


Figure 4.8.5: Deux positions dans l'espace de redondance défini par $\boldsymbol{\alpha} = (\beta(rad), a_{tcp}(rad), x_b(m), y_b(m), \theta_b(rad))$ pour une tâche de positionnement donnée.

Figure 4.9.1: Ordre de grandeur des rigidités pour différents systèmes [42].

4.9 Critères de performance pour tâches statiques pour la gestion de redondance

Des machines à commande numérique, la production automatisée tends à augmenter la versatilité des systèmes robotiques. Des systèmes de taille de plus en plus réduite peuplent les espaces de production pour permettre des solutions sécuritaires, agiles et reconfigurables. Pour étendre leur champ d'action, les robots se dotent aujourd'hui de capteurs leur servant à détecter ou à anticiper l'occurrence de collisions. Ceci leur permet, entre autres choses, de travailler dans des milieux peuplés d'humains. L'utilisation de ces robots collaboratifs, tels que le LBR iiwa, pour des opérations industrielles normalement effectuées par des robots gros porteurs est cependant jalonnée de défis. La grande flexibilité mécanique de ces systèmes (Fig. 4.9.1) engendre des imprécisions sur le placement de l'effecteur du système, notamment lorsqu'une interaction physique a lieu avec l'environnement. En outre, la limitation des couples articulaires est aussi un obstacle à la réalisation de certaines tâches nécessitant des interactions fortes avec l'environnement physique. Ces deux aspects, que sont l'imprécision due à l'application d'un effort sur l'effecteur, et la capacité d'application d'effort du

robot, sont abordés dans cette thèse.

4.9.1 Optimisation du comportement en déformation sous l'influence d'une force statique

On s'intéresse dans un premier temps aux imprécisions statiques de l'effecteur d'un robot, sous l'influence d'une force extérieure. Pour cela, on modélise le robot comme un assemblage de solides rigides (les segments) liés entre eux par des ressorts torsionnels à caractéristique linéaire Fig. 4.9.2, situés au niveau des articulations. Ce modèle permet de calculer la matrice de compliance Cartésienne du robot $\boldsymbol{C}$.

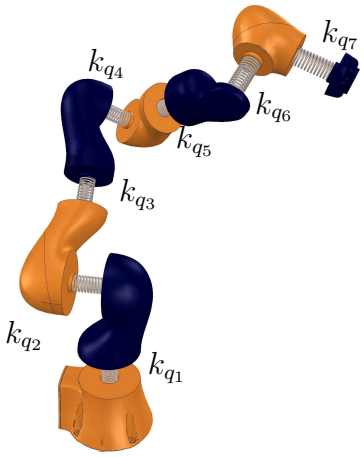


Figure 4.9.2: Modèle élastique d'un LBR iiwa concentrant la flexibilité dans les articulations (ressorts torsionnels linéaires).

$$\boldsymbol{\delta}\mathbf{X} = \boldsymbol{C}\mathbf{f} = (\boldsymbol{J}\boldsymbol{K}_q{}^{-1}\boldsymbol{J}^{\intercal})\mathbf{f}. \tag{4.27}$$

Avec :

$$\boldsymbol{\delta}\mathbf{X} = \begin{bmatrix} \boldsymbol{\delta}\mathbf{p} \\ \boldsymbol{\omega} \end{bmatrix} \text{ et } \mathbf{f} = \begin{bmatrix} \mathbf{F} \\ \mathbf{M} \end{bmatrix}.$$

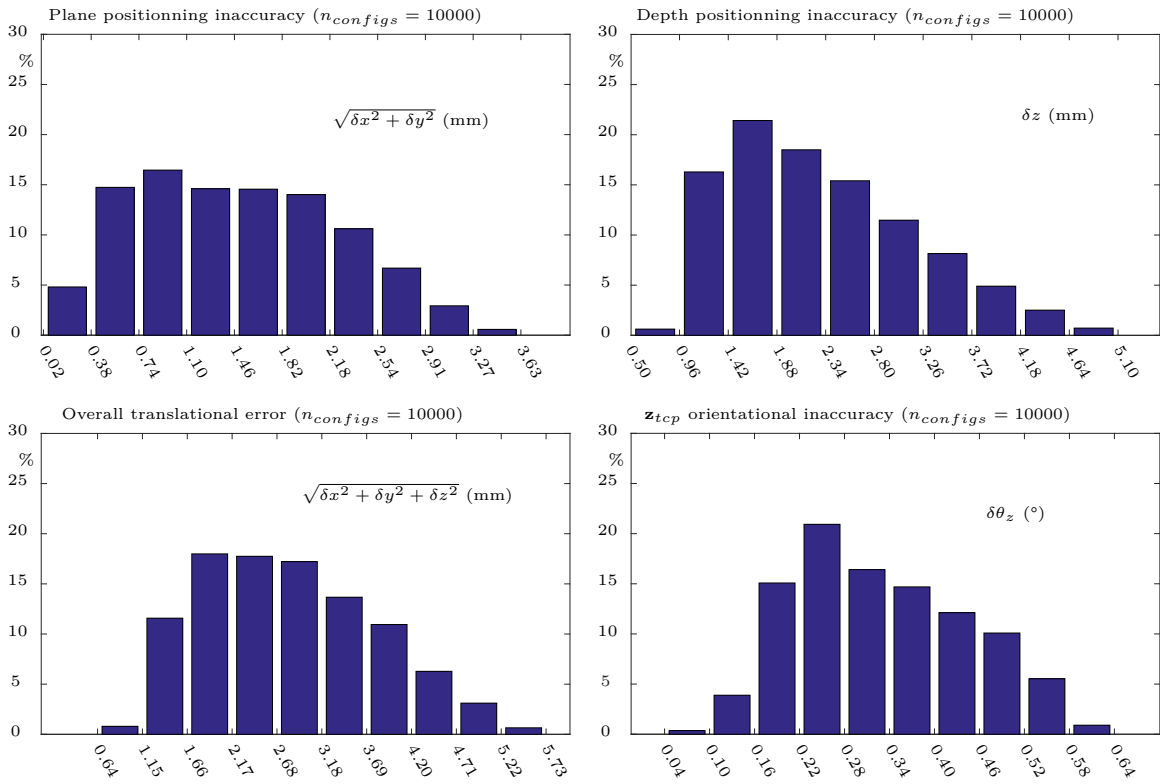


Figure 4.9.3: Étude des imprécisions de placement du TCP sous l'influence d'un effort statique ${}^{tcp}\mathbf{f} = \begin{bmatrix} 0(N) & 0(N) & -100(N) & 0(Nm) & 0(Nm) & -5(Nm) \end{bmatrix}^\intercal$ pour 10000 configurations articulaires échantillonnées aléatoirement.

La compliance Cartésienne du robot lie directement les efforts statiques appliqués à l'effecteur, aux déplacements (imprécisions liées à la flexibilité mécanique) de l'effecteur, pour une configuration articulaire donnée.

On peut définir un certain nombre de critères d'imprécision. Le vecteur de déplacement issu de Eq. (4.27) est divisé en une composante translationnelle et une composante rotationnelle que l'on peut exploiter séparément.

Pour le LBR iiwa, des simulations sur 10000 configurations articulaires et un effecteur de taille classique montrent que les déplacements Cartésiens de cet effecteurs

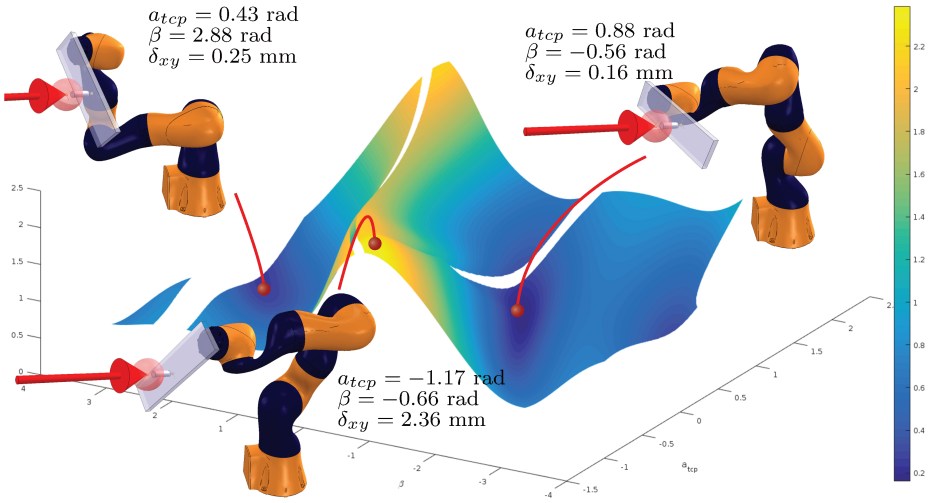


Figure 4.9.4: Déplacement dans le plan $x_{tcp} - y_{tcp}$ pour un effort statique ${}^{tcp}\mathbf{f} = \begin{bmatrix} 0(N) & 0(N) & -100(N) & 0(Nm) & 0(Nm) & -5(Nm) \end{bmatrix}^{\mathsf{T}}$ appliqué au TCP en fonction des deux paramètres de redondance β (angle de coude) et a_{tcp} (orientation de l'outil autour de z_{tcp}).

sont relativement importants (Fig. 4.9.3).

La redondance de positionnement de l'iiwa, vis à vis d'une tâche à symétrie de révolution, consiste en une redondance lié à l'angle de coude β (Fig. 4.8.2) et une redondance d'orientation de l'effecteur autour de l'axe de symétrie de révolution a_{tcp} (Fig. 4.8.3). L'évolution du critère d'imprécision de positionnement dans le plan ($\delta_{xy} = \sqrt{\delta x^2 + \delta y^2}$) dans cet espace de redondance est visualisable en Fig. 4.9.4. On peut y observer les postures du robot associées à 3 positions dans cet espace de redondance pour lesquelles l'imprécision est maximale ou minimale vis à vis de l'effort d'entrée.

On peut visualiser en parallèle l'évolution de plusieurs critères de précision pour la même tâche (Fig. 4.9.5), ce qui permet de sélectionner une position dans l'espace de redondance qui puisse limiter l'imprécision pour tous ces critères en même temps.

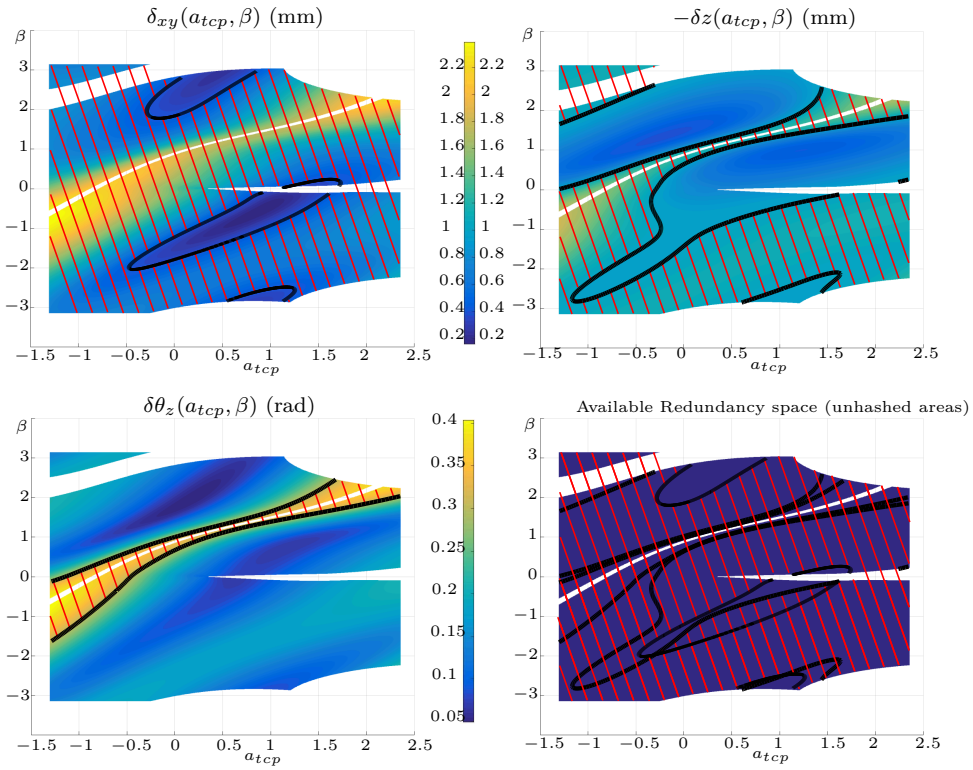


Figure 4.9.5: Quelques mesures d'imprécision du TCP pour un effort statique ${}^{tcp}\mathbf{f} = \begin{bmatrix}0(N) & 0(N) & -100(N) & 0(Nm) & 0(Nm) & -5(Nm)\end{bmatrix}^\intercal$ appliqué au TCP en fonction des deux paramètres de redondance β (angle de coude) et a_{tcp} (orientation de l'outil autour de z_{tcp}). La tolérance pour le déplacement dans le plan $\delta_{xy} = \sqrt{\delta x^2 + \delta y^2}$ est $\delta_{xy_{max}} = 0.4$, celle pour le déplacement dans la direction indiquée par z_{tcp}, δ_z est $\delta_{z_{max}} = 1$ mm, celle pour le défaut d'orientation de l'axe z_{tcp}, $\delta\theta_z$ est $\delta_{\theta_{max}} = 0.3°$. Un ensemble de positions de l'espace de redondance peut être sélectionné pour se conformer à toutes les tolérances décrites ci-dessus (figure en bas à droite).

4.9.2 Optimisation de la capacité d'application d'effort

De la même manière, on peut s'intéresser à la capacité qu'a un robot sériel à contrer un effort extérieur donné. Cette problématique est intéressante dans la mesure ou la puissance (et donc les couples articulaires) des robots collaboratifs est bridée pour des raisons de sécurité, alors que les procédés (tels que le perçage, l'usinage, ou le rivetage) nécessitent toujours la même intensité d'interaction outil-matière. Dans le contexte de l'utilisation d'un système redondant vis à vis d'une tâche de positionnement, il peut être intéressant d'observer les variations de cette capacité d'application d'effort pour l'ensemble des solutions articulaires disponibles en se conformant à la réalisation de cette tâche.

La définition que nous exploitons et développons dans cette thèse, est celle de l'index de capacité d'application d'effort (ou FCI pour Force Capacity Index). Le FCI correspond, pour une configuration articulaire $\mathbf{q}$ donnée, une action mécanique $-\mathbf{f}$ donnée et un TCP donné, au multiplicateur (réel) minimal $\lambda_{sat} \in \mathbb{R}^+$ de l'action mécanique qui fait saturer en effort au moins un des actionneurs du robot. Ce multiplicateur correspond aussi à l'intensité maximale de l'action mécanique qui est soutenable pour le système.

Dans une configuration $\mathbf{q}$ donnée et pour une action mécanique $-\lambda\mathbf{f}$ ($\lambda \in \mathbb{R}^+$) à contrer au niveau de l'effecteur, les n actionneurs d'un robot sériel doivent produire chacun un couple τ_i que l'on peut synthétiser dans un vecteur $\boldsymbol{\tau} = [\tau_1 \ \dots \tau_n]^\intercal$. Ce vecteur est la somme $\boldsymbol{\tau} = \boldsymbol{\tau}_\mathbf{g} + \boldsymbol{\tau}_{\lambda\mathbf{f}}$ d'un vecteur $\boldsymbol{\tau}_\mathbf{g} = [\tau_{\mathbf{g},1} \ \dots \tau_{\mathbf{g},n}]^\intercal$ regroupant les couples nécessaires pour lutter contre l'effet gravitationnel dans la configuration $\mathbf{q}$, et d'un vecteur $\boldsymbol{\tau}_{\lambda\mathbf{f}} = [\tau_{\lambda\mathbf{f},1} \ \dots \tau_{\lambda\mathbf{f},n}]^\intercal$ regroupant les couples nécessaires pour lutter contre une force $-\lambda\mathbf{f}$ au niveau de l'effecteur, dans la configuration $\mathbf{q}$. Cette somme peut aussi s'écrire $\boldsymbol{\tau} = \boldsymbol{\tau}_\mathbf{g} + \lambda\boldsymbol{\tau}_\mathbf{f}$. Une représentation de ces couples est visible en Fig. 4.9.6 pour un robot à 3 degrés de liberté. Le polytope de couples articulaires correspond au volume contenant l'ensemble des vecteurs couples que peut produire le robot (l'actionneur i produit un couple $\tau_i \in [\tau_{i,l}, \tau_{i,u}]$).

Le FCI peut être calculé en déterminant, pour chaque actionneur i, l'intensité λ_i de

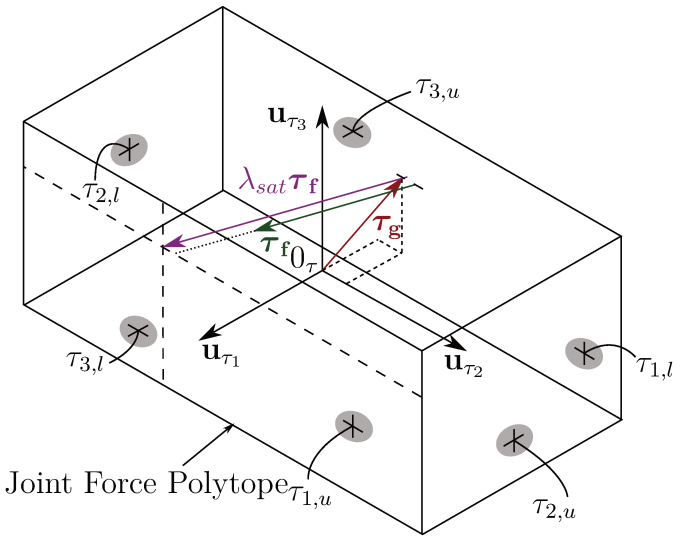


Figure 4.9.6: Polytope des efforts articulaires pour un robot 3 axes avec des limites de couples $\tau_{i,l}$ and $\tau_{i,u}$ for $i \in [\![1,3]\!]$, et définition de l'intensité d'effort saturant λ_{sat}.

l'effort $\mathbf{f}$ qui le fera saturer en couple (*i.e.* $\lambda_i * \tau_{\mathbf{f},i} + \tau_{\mathbf{g},i} = \tau_{i,l}$ ou $\lambda_i * \tau_{\mathbf{f},i} + \tau_{\mathbf{g},i} = \tau_{i,u}$). Le FCI correspond au minimum de ces intensités. Sa définition exacte est :

$$\forall i \in [\![1,n]\!], \lambda_i = \begin{cases} \frac{\tau_{u,i} - \tau_{\mathbf{g},i}}{\tau_{\mathbf{f},i}}, \text{ si } \tau_{\mathbf{f},i} > 0 \\ \frac{\tau_{l,i} - \tau_{\mathbf{g},i}}{\tau_{\mathbf{f},i}}, \text{ si } \tau_{\mathbf{f},i} < 0 \\ \infty, \text{ if } \tau_{\mathbf{f},i} = 0 \end{cases}$$

$$\lambda_{sat} = min((\lambda_i)_{i \in [\![1,n]\!]})$$

Pour un robot plan à 3 degrés de liberté, devant réaliser une tâche de positionnement de l'effecteur en un point $M = (x, y)$ du plan tout en contrant un effort $\mathbf{f}$, l'intensité saturante de chaque axe peut être tracée en fonction du paramètre de redondance définit par l'angle θ (*i.e.* angle défini par la direction prise par dernier segment du robot). Ce tracé est effectué en Fig. 4.9.7.

Le FCI peut également être tracé pour le LBR iiwa, et pour un espace de redondance

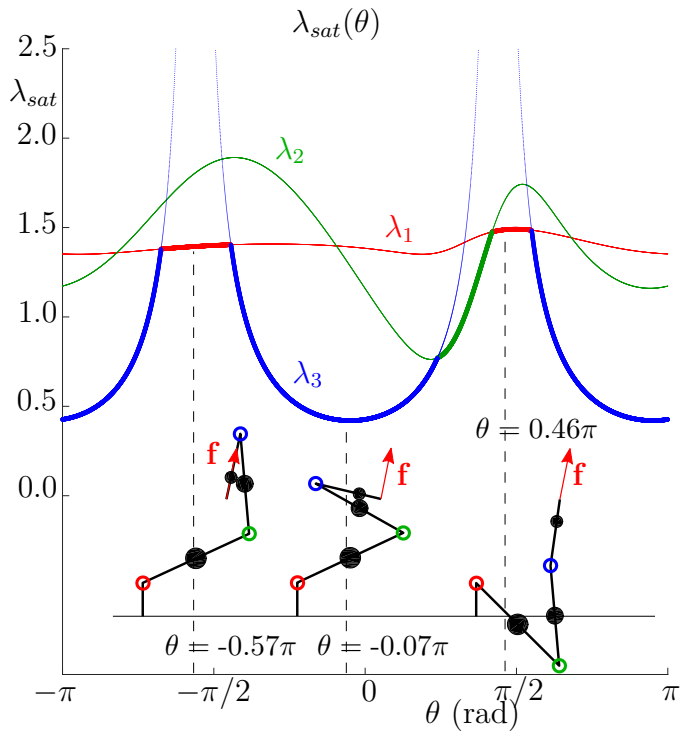


Figure 4.9.7: Intensité saturantes des actionneurs λ_1, λ_2 and λ_3 (traits fins) et intensité saturante λ_{sat} (trait épais) d'un robot plan à 3 degrés de liberté effectuant une tâche de positionnement 2D de son organe terminal (induisant une redondance paramétrable par $\theta \in [-\pi, \pi]$) et contrant un effort $-\mathbf{f}$.

à 2 paramètres. La redondance de positionnement de l'iiwa, vis à vis d'une tâche à symétrie de révolution, consiste en une redondance lié à l'angle de coude β (Fig. 4.8.2) et une redondance d'orientation de l'effecteur autour de l'axe de symétrie de révolution a_{tcp} (Fig. 4.8.3). Les variations du critère de capacité d'effort λ_{sat} sont visibles dans la Fig. 4.9.8. On peut y observer les postures du robot associée à 3 positions dans cet espace de redondance pour lesquelles la capacité d'application d'effort est minimale ou maximale.

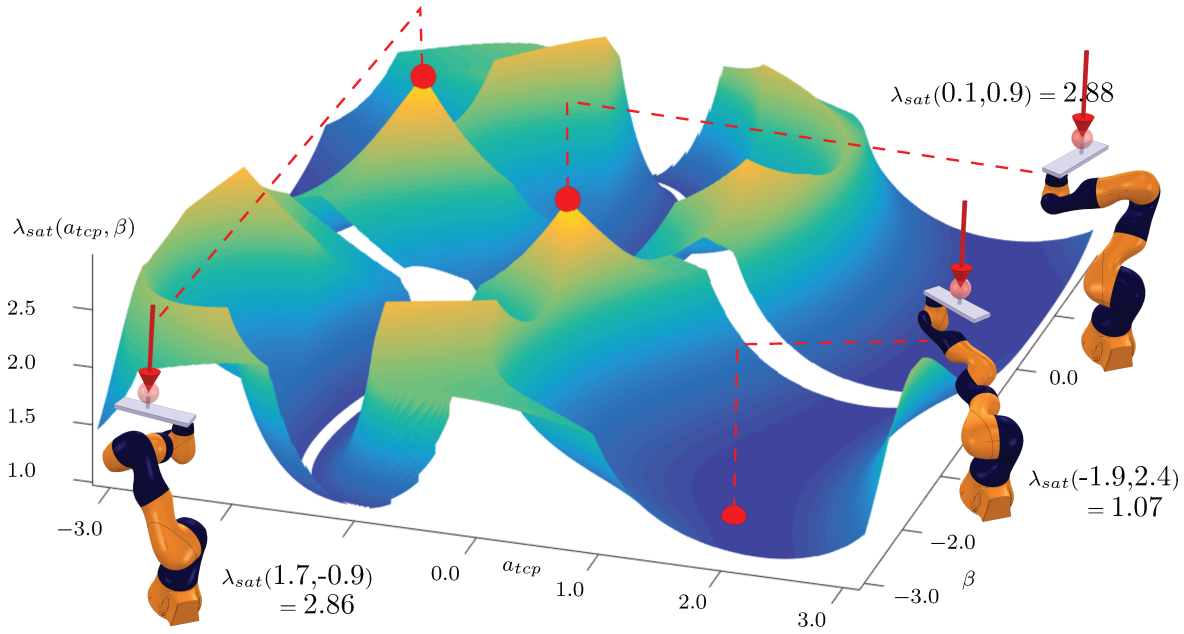


Figure 4.9.8: Intensité d'effort saturante d'un KUKA LBR iiwa R820, doté d'un effecteur pesant 5.0 kg, pour un effort statique représenté par la flèche rouge.

4.10 Une approche de planification de mouvement en environnement dynamique : RH-DRM

4.10.1 Introduction

Le sujet de la planification de mouvements (motion planning), en robotique, se réfère aux méthodes permettant de trouver des trajectoires sans collision pour des systèmes cinématiques. Ces méthodes se basent sur des représentations de la topologie de l'espace et du système cinématique, pour planifier des trajectoires admissibles entre un point de départ et une destination.

Dans l'industrie de la robotique, le problème de la planification de mouvements des systèmes robotisés est traditionnellement effectué lors d'une phase de programmation hors ligne (PHL). La topologie de l'environnement du robot, dans ces méthodes de PHL, est toujours structurée, prévisible voire absolument immobile. En effet, les

robots sont traditionnellement amenés à fonctionner dans des enceintes fermées, où la moindre intrusion humaine doit les stopper immédiatement. Un des objectif de l'usine agile est d'apporter de la modularité, de la reconfigurabilité aux systèmes productiques. Cette flexibilité n'est possible qu'en augmentant l'adaptabilité, et donc l'autonomie de ces systèmes. L'arrivée sur la scène industrielle des robots collaboratifs est inscrite dans ce désir d'agilité. Cependant, la présence d'humains autour des cobots multiplie à l'infinie la variété des topologies environnementales possibles. La programmation hors ligne des robots, qui est le moyen traditionnel de planification des mouvements de ces systèmes, n'est pas un outil adéquat dans cette recherche de flexibilité, car elle est incapable de prédire toutes les topologies possibles de l'espace dans lequel évolueraient les systèmes, et encore moins de planifier des trajectoires pour toutes les situations pouvant survenir.

L'enjeu de cette partie est de présenter une stratégie de planification de mouvements, appelée RH-DRM (Receding Horizon Dynamic RoadMaps) qui soit adaptée aux environnements dynamiques. Cette approche se base sur les Dynamic Roadmaps (DRM) [139], elles-mêmes basées sur les Probabilistic Roadmaps (PRM) [121], ainsi que sur une adaptation de l'algorithme A* [135] pour les graphes à intervalles. Les PRM et ses déclinaisons font partie de la grande famille des méthodes basées sur l'échantillonnage de l'espace des configurations (sampling based methods), qui comptent aujourd'hui parmi les méthodes les plus populaires de planification de mouvements.

4.10.2 Présentation des outils à la base du RH-DRM

4.10.2.1 PRM et DRM

Le principe de base des PRM est d'une part un apprentissage discret des mouvements libres dans l'espace articulaire (phase hors ligne), et d'autre part, l'exploitation de cet apprentissage pour générer des trajectoires entre des configurations libres (phase en ligne). La phase d'apprentissage correspond à la création d'un graphe de connectivité de l'espace des configurations du robot. Le graphe de connectivité se présente sous la

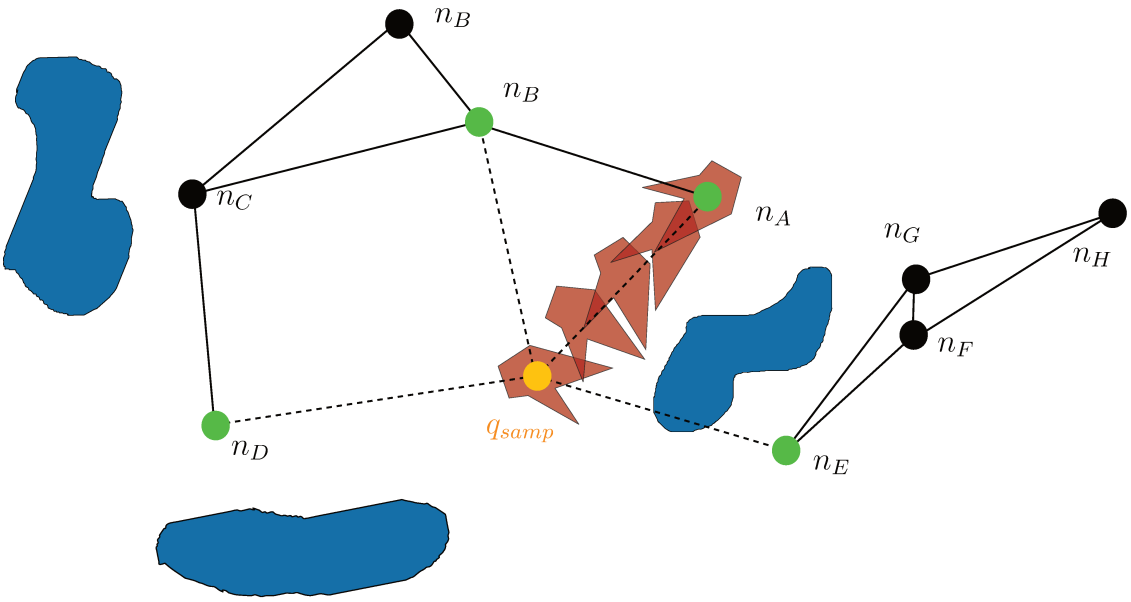


Figure 4.10.1: Insertion d'un nœud et d'un segment pendant la construction du graphe de connectivité. Le nouveau nœud est associé à la configuration q_samp, et des raccordements vont être tentés avec les nœuds n_D, n_B, n_A et $n - E$. Pour tester les collisions entre q_samp et n_A, la trajectoire linéaire entre ces deux configurations est échantillonnée, l'enveloppe de l'agent (rouge) y est simulée et des tests de collisions avec les obstacles (bleus) sont réalisés.

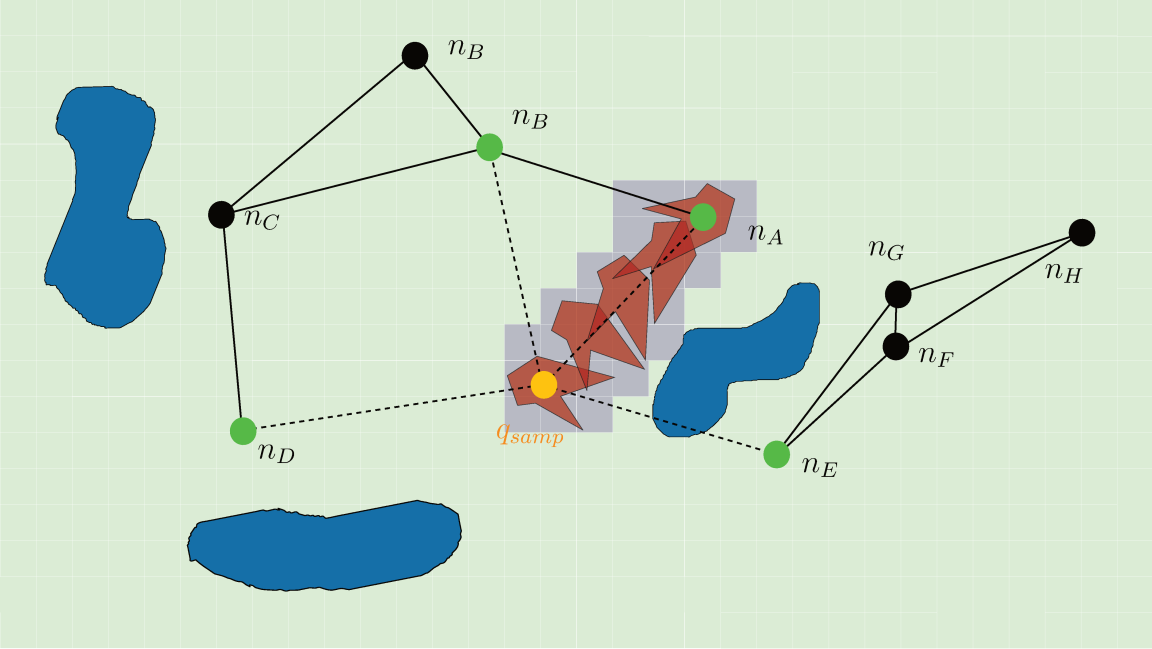


Figure 4.10.2: Procédure d'insertion d'un segment dans le graphe de connectivité, pendant sa phase de construction.

forme d'un ensemble de nœuds correspondant à des configurations articulaires dans lesquelles le robot n'est pas en collision avec son environnement statique. Les segments du graphe correspondent à des trajectoires linéaires entre les nœuds (*i.e.* les configurations) qu'ils relient, et qui ne présentent pas de collision avec l'environnement. Cette phase peut être relativement longue, car de nombreuses vérifications de collisions entre solides sont effectuées. Le principe de cette étape d'apprentissage, et notamment l'insertion d'un nœud et d'un segment dans le graphe de connectivité, est illustrée en Fig. 4.10.1. Ce graphe servira ensuite en ligne pour relier de manière continue, et sans collision, des configurations de départ à des configurations d'arrivée. Comme il présente de la cyclicité, il existe souvent plusieurs chemins permettant de relier des nœuds entre eux. Le problème de résolution du chemin le plus court (shortest path problem) est résolu par des méthodes bien connues comme Djikstra, ou A*.

Dans le contexte étudié dans cette thèse, l'environnement est dynamique. Le graphe

de connectivité est donc amené à changer au cours du temps pour s'adapter aux obstacles mobiles. Les DRM (voir Algo. 7) permettent de rendre compte de ces changements en modifiant efficacement la topologie du graphe de connectivité en fonction de l'occupation de l'espace par des obstacles mobiles. Ceci est rendu possible en créant pendant la phase hors ligne, en parallèle de la création du graphe de connectivité, un mapping entre l'espace occupé par le robot dans chaque nœud et segment du graphe. Ce mapping se présente sous la forme d'une grille d'occupation de l'espace pour laquelle chaque élément (pixel ou voxel) contient les références de nœuds ou de segments du graphe qui les concernent directement. Dans une configuration associée à un nœud (ou à un segment), si l'enveloppe externe du robot occupe une partie d'un pixel/voxel alors la référence de ce nœud ou de ce segment est associé au pixel/voxel. Une illustration de ce processus est visible en Fig. 4.10.2. Lors de la phase en ligne, lorsqu'un obstacle dynamique occupe partiellement un pixel/voxel, tous les nœuds et segments associés à ce pixel/voxel sont retirés temporairement du graphe de connectivité. Ceci permet de ne conserver que des nœuds et segments pour lesquels aucune collision avec l'obstacle n'est possible. Le calcul du chemin le plus court est effectué sur cette version dégradée du graphe de connectivité.

4.10.2.2 A*

Lors de la phase en ligne, le calcul du chemin le plus court dans le graphe doit être effectué très rapidement. Une méthode connue pour la recherche de chemin dans les graphes pondérés statique est l'algorithme A* (Algo. 8). Cette méthode dérive directement de la méthode de Dijkstra (recherche uniforme), qui consiste à explorer le graphe de proche en proche à partir du nœud de départ, jusqu'à arriver au nœud de destination. Dans cette méthode, à chaque nœud est associé le coût du chemin le plus court trouvé pour s'y rendre à partir du nœud de départ. Ce coût est contenu dans une variable appelée g. Initialement, tous les nœuds ont une valeur de g infinie sauf le nœud de départ, pour qui $g = 0$. à chaque itération de l'algorithme, le nœud ayant le plus petit score de g est sélectionné, et l'algorithme procède à son "expansion". La phase d'expansion consiste à mettre à jour la valeur de g des voisins

inputs :

- scGeom // The static scene geometry
- roGeom // The robot geometry
- occGrid // An occupancy grid
- nNodes // The number of nodes
- kNeighs // The number of connexions sought from each node

output: A "Dynamic RoadMap" consisting in :

- $\mathcal{G}(\mathcal{N},\mathcal{E})$ // a graph ($\mathcal{G}$) comprised of nodes ($\mathcal{N}$) and edges ($\mathcal{E}$)
- occNodes // mapping between $\mathcal{N}$ and occGrid
- occEdges // mapping between $\mathcal{E}$ and occGrid

```
1  Procedure contructDRM(scGeom, roGeom, occGrid, nNodes, kNeighs)
2      N ← ∅ ;
3      E ← ∅ ;
4      occNodes ← ∅ ;
5      occEdges ← ∅ ;
6      configs ← generateFreeConfigs(nNodes, scGeom, roGeom) ;
7      N ← createNodes(configs);
8      foreach c ∈ N do
9          occupiedPixels ← pixelsOccupiedBy(c, roGeom, occGrid) ;
10         linkNodeAndPixels(occNodes,c, occupiedPixels) ;
11         cNeighs ← getKNearestNeighbours(c,N \ {c},kNeighs);
12         foreach n ∈ cNeighs, that has not yet been connected to c do
13             edgeCN ← createEdgeConnecting(c,n);
14             [isEdgeFree, sweptPixels] ← sweepEdge(edgeCN, scGeom, roGeom,
                occGrid);
15             if isEdgeFree then
16                 E ← E ∪ {edgeCN} ;
17                 linkEdgeAndPixels(occEdges,edgeCN, sweptPixels) ;
18             end
19         end
20     end
21     G ← G(N,E) ;
22     return [G, occNodes, occEdges] ;
23 end
```

Algorithm 7: Construction d'une Dynamic roadmap (DRM). Les modifications par rapport à l'algorithme de construction du PRM (Algo. 2) sont montrés en bleu foncé.

du nœud sélectionné pour lesquels le chemin démarrant au nœud de départ, passant par le nœud sélectionné, et allant jusqu'à eux est plus court (*i.e.* a un coût plus faible). L'algorithme s'arrête quand le nœud de destination est sélectionné. Cette méthode, également appelée recherche uniforme, explore le graphe, comme son nom l'indique, de manière uniforme. Elle étend la recherche de manière uniforme autour du nœud de départ, en explorant toutes les options de chemins existants prenant racine au nœud de départ, et ayant un coût similaire. L'algorithme A*, en revanche, biaise l'étape de sélection en choisissant le nœud non pas uniquement en fonction de sa distance au nœud de départ, mais également en fonction d'un coût estimé pour rejoindre le nœud de destination. Cette valeur estimée du coût pour rejoindre le nœud de destination est appelée valeur d'heuristique; celle-ci est calculée à l'étape d'initialisation de l'algorithme et est stockée dans la variable h.

input :

- $\mathcal{G}(\mathcal{N}, \mathcal{E})$ // a graph ($\mathcal{G}$) comprised of nodes ($\mathcal{N}$) and edges ($\mathcal{E}$)
- sStart // the reference to the starting node
- sEnd // the reference to the destination node

output: path // an ordered list of node references

```
1  Procedure AStar(G(N,E), sStart, sEnd)
2      forall s ∈ N do
3          s.g ← ∞;
4          s.bp ← ∅;
5          s.h ← heuristicDistanceBetween(s, sEnd);
6      end
7      sStart.g ← 0;
8      open ← ∅, closed ← ∅;
9      add(sStart,open);
10     loop
11         s ← pop(open);
12         if s = sEnd then break loop ;
13         else if s = ∅ then return ∅ ;
14         foreach n ∈ neighbours(s,E), n ∉ closed do
15             costToNThroughS ← s.g + cost(s,n,E);
16             if costToNThroughS < n.g then
17                 n.g ← costToNThroughS;
18                 n.bp ← s;
19                 if n ∉ open then add(n,open) ;
20             end
21         end
22         remove(s,open),add(s,closed);
23     end
24     s ← sEnd;
25     rpath ← ∅, append(s, rpath);
26     while s ≠ sStart do
27         p ← s.bp;
28         s ← p;
29         append(s, rpath);
30     end
31     path ← reverse(rpath);
32     return path;
33 end
34 Procedure pop(open)
35     s ← arg min_{c∈open}(c.g + c.h) ;
36     return s;
37 end
```

Algorithm 8: L'algorithme A* pour la recherche de chemin dans un graphe pondéré.

4.10.3 Le besoin d'anticipation dans la planification de mouvements

Les DRM fournissent un moyen efficace de réagir à des changements dans l'environnement. Par rapport aux PRM, la phase hors ligne de construction du graphe de connectivité est plus onéreuse. Elle requiert des tests de collisions avec chaque pixel/voxel pour chaque nœud et segment du graphe. Fort heureusement, cette phase se déroule hors ligne, et n'affecte que très peu l'efficacité de l'algorithme quand il est réellement utilisé. La phase en ligne du DRM, par rapport à celle du PRM, est peu impactée, une fois que l'occupation des obstacles est déterminée. L'opération consistant à rendre indisponibles les nœuds et segments du graphe concernés par les obstacles est très simple et rapide. La stratégie du DRM est donc très intéressante. Elle permet de prendre en compte les obstacles mobiles pendant la phase de planification sans augmenter de beaucoup le coût de l'algorithme. Elle conserve tous les avantages des PRM, et permet plus.

Une limitation majeure du cadre des DRM, cependant, est le fait que les chemins planifiés ne prennent en compte l'occupation des obstacles qu'à un instant unique, et ignorent leur mouvement. Cela revient à planifier des chemins comme si les obstacles dynamiques restaient immobiles. Les configurations futures du système sont planifiées sans prendre en compte l'occupation future de l'espace par les obstacles.

Comme pour beaucoup de méthodes basées sur l'échantillonnage de l'espace des configurations, l'approche des DRM est globale vis à vis de l'espace. Elle a la capacité de planifier des chemins globalement (d'un point de vue spatial) au lieu de ne réagir que par rapport à ce qui se trouve dans l'environnement proche. Cependant, le cadre des DRM planifie localement vis à vis du temps, puisqu'il ne regarde que l'occupation actuelle (*i.e.* locale vis à vis du temps) des obstacles. Cette hypothèse de ne prendre en compte l'aspect temporel que de manière locale est justifiée si la vitesse de l'agent est très grande par rapport à celle des obstacles. Cependant, dans notre cas, les obstacles mobiles sont des humains ou des homologues robotiques. Par conséquent, les vitesses impliquées sont relativement proches les unes des autres, et les obstacles ne peuvent être considérés comme immobiles.

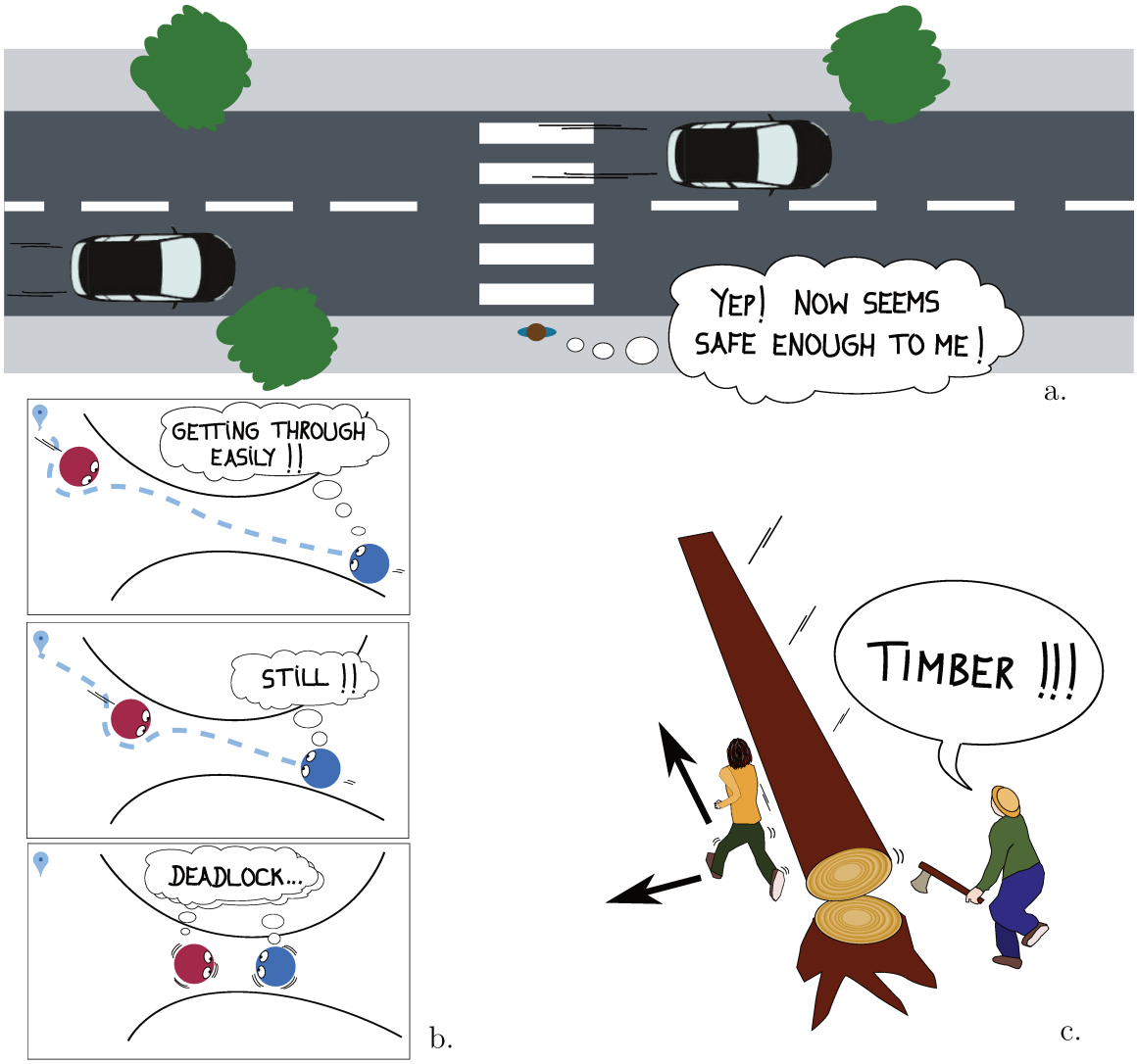


Figure 4.10.3: Exemples illustrant le besoin d'anticipation des mouvements environnants.

La Fig. 4.10.3 illustre des situations dans lesquelles l'anticipation est nécessaire mais pas utilisée pour planifier des mouvements. Ces exemples suggèrent ce que l'utilisation des dynamic roadmaps dans des environnements dynamiques pourrait provoquer. Des situations de coincements (deadlocks) pourraient facilement survenir si les mouvements étaient planifiés sans prendre en compte le mouvement des obstacles. Des situations dangereuses pourraient aussi survenir, pouvant provoquer des collisions, de la casse ou des blessures.

En règle générale, l'anticipation devient obligatoire pour la planification de mouvements dès lors que les mouvements des obstacles ont une vitesse relative non négligeable par rapport à celle de l'agent. La prédiction des mouvements permet même de planifier des trajectoires dans des environnements pour lesquels les obstacles ont des vitesses plus importantes que l'agent lui-même.

4.10.4 Receding Horizon Dynamic RoadMaps (RH-DRM)

La stratégie présentée ici est appelée Receding Horizon Dynamic RoadMaps (RH-DRM). Elle étend le concept des DRM en utilisant une anticipation de l'occupation de l'espace sur un horizon de temps. Les avantages des DRM sont conservés. La méthode est générique aux les robots mobiles ou/et sériels, utilise toujours une représentation discrète permettant de lier l'espace d'évolution des obstacles et celui des configurations, est efficace et obéit à des règles simples et sensées. Cette nouvelle stratégie planifie un trajet (chemin géométrique et timing) permettant d'arriver au plus tôt à la destination sans entrer en collision avec des obstacles dynamiques (dont la trajectoire est estimée). Le trajet peut donc potentiellement admettre des arrêts ainsi que des passages multiples dans une configuration, ce qui est impossible avec les algorithmes présentés ci-avant.

La phase hors ligne des RH-DRM est exactement la même que celle des DRM. le graphe de connectivité est construit en parallèle de la grille d'occupation, et des liens qui existent entre chaque élément de cette grille et les configurations associées au nœuds/segment du graphe. A partir d'une anticipation en ligne des mouvements des obstacles sur un horizon de temps futur, l'anticipation d'occupation des pixels/voxels

est calculée, ce qui permet enfin de prédire la topologie du graphe de connectivité.

Les RH-DRM se basent sur une représentation particulière du graphe de connectivité, dont la topologie est variable et pour lesquels la traversée d'un segment n'est pas immédiate. Un type de graphe baptisé dans ce rapport "graphe à intervalles de début de traversée" (ou "step-in interval graph", en anglais), est utilisé pour représenter de manière succincte le problème (voir Section 4.10.4.1).

En utilisant ce graphe, l'objectif va être de trouver le chemin permettant d'arriver au plus tôt à un nœud de destination, à partir d'un nœud de départ. Cette recherche de chemin est effectuée grâce à un algorithme inspiré de la méthode A*.

4.10.4.1 Graphes à intervalles de début de traversée

Un période de disponibilité d'un nœud/segment garantit qu'aucune collision n'aura lieu sur ce nœud/segment pendant cet intervalle de temps (voir Fig. 4.10.4 a)). La décision d'emprunter ou non un segment doit garantir la disponibilité du segment pendant toute sa traversée. Par conséquent, on peut commencer à traverser un segment entre l'instant où ce segment devient disponible, et le dernier instant auquel il est disponible auquel on a retranché la durée de traversée. Cette nouvelle disponibilité est baptisée "disponibilité de début de traversée". Elle est à la base de la construction des graphes à intervalles de début de traversée (voir Fig. 4.10.4 b)). On modifie donc les intervalles de disponibilité des segments pour prendre en compte cette contrainte.

Une représentation compacte des graphes à intervalles de début de traversée est un graphe $\mathcal{G}(\mathcal{N}, \mathcal{E}, \mathcal{D})$, composé d'un ensemble $\mathcal{N}$ des nœuds du graphe, d'un ensemble $\mathcal{E}$ des segments du graphe, et d'un ensemble $\mathcal{D} = \{t_0, \ldots, t_{N_0}\}$ des dates de changement de connectivité de début de traversée. Chaque nœud $n \in \mathcal{N}$ est un N_0-uplet logique regroupant la disponibilité du nœud pendant tous les intervalles $[t_i, t_{i+1}[$, $i \in [\![0..N_0 - 1]\!]$. Chaque segment $e \in \mathcal{E}$, en plus de contenir les références des nœuds qu'il lie, est aussi un N_0-uplet logique regroupant la disponibilité du nœud pendant tous les intervalles $[t_i, t_{i+1}[$, $i \in [\![0..N_0 - 1]\!]$.

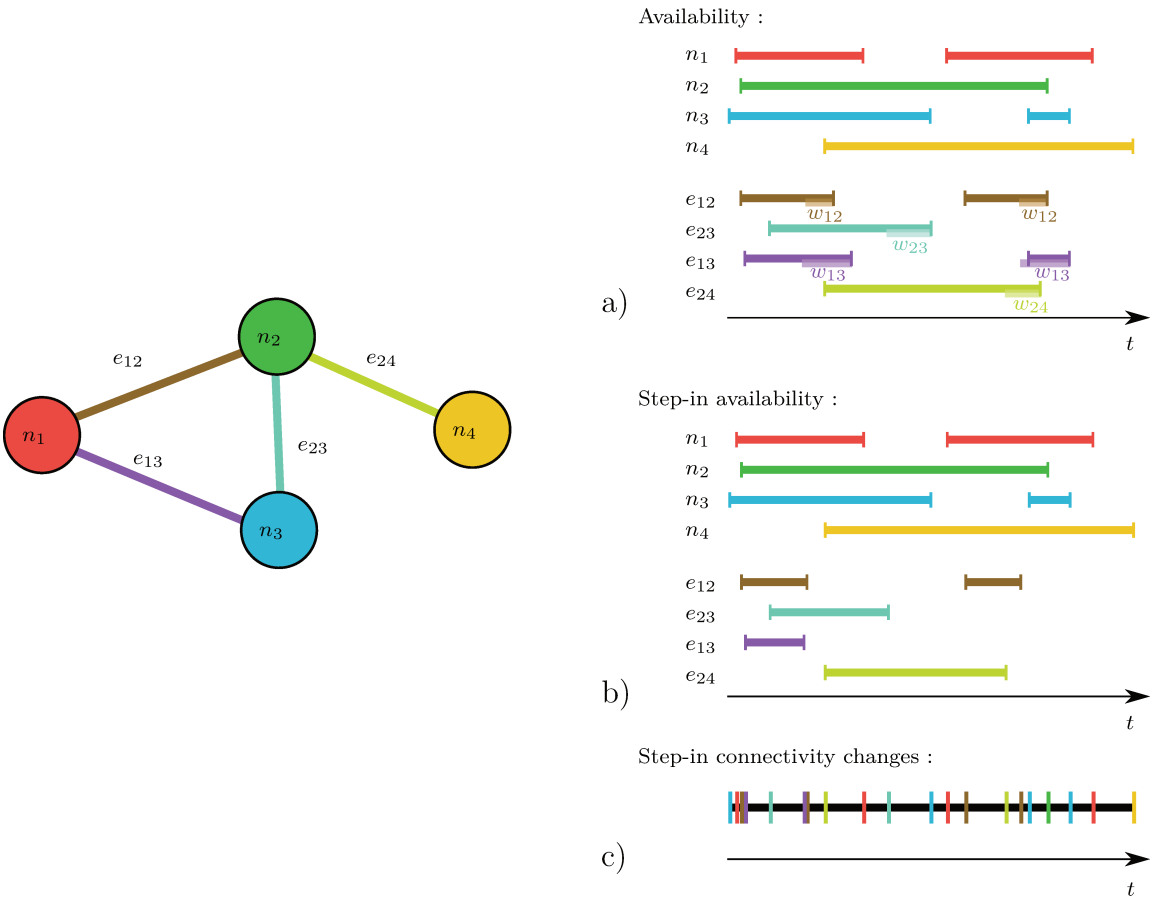


Figure 4.10.4: Un graphe à 4 nœuds $(n_i)_{i\in[\![1..4]\!]}$ et 4 segments $(e_{ij})_{i,j\in[\![1..4]\!]^2}$. La couleur de représentation de ces nœuds/segments est utilisée pour représenter en a) et b) les intervalles de temps pendant lesquels ils sont disponibles.
a) Une représentation d'un graphe à intervalles, qui montre la disponibilité des nœuds $(n_i)_{i\in[\![1..4]\!]}$ et des segments $(e_{ij})_{i,j\in[\![1..4]\!]^2}$. Les poids des segments (qui sont aussi les durées de traversée) sont les $(w_{ij})_{i,j\in[\![1..4]\!]^2}$ correspondant.
b) Une représentation des graphes à intervalles de début de traversée. La disponibilité des segments est raccourcie du temps nécessaire pour les traverser, qui correspond à leur poids $(w_{ij})_{i,j\in[\![1..4]\!]^2}$.
c) Des changements dans la disponibilité de début de traversée des segments du graphe (step-in connectivity) apparaissent à chaque marqueur vertical montré ici. La connectivité de début de traversée reste constante entre deux marqueurs successifs.

Une représentation équivalente que nous préférons pour notre explication est une succession de N_0 graphes statiques, qui sont valables chacun pendant un intervalle de temps $[t_i, t_{i+1}[$, $i \in [\![0..N_0 - 1]\!]$ différent. Chaque nœud de ces graphes correspond à une instance temporelle des nœuds de $\mathcal{N}$, et sera valable uniquement pendant un intervalle $[t_i, t_{i+1}[$. On appellera ces objets des *instances temporelles de nœuds (ITN)*. Chacun de ces graphes peut être représenté par une matrice d'adjacence de début de traversée $\boldsymbol{A}^{[t_i,t_{i+1}[}$. Cette matrice décrit la connectivité de début de traversée de chaque nœud et segment pendant l'intervalle $[t_i, t_{i+1}[$. Les termes $a_{k,l}^{[t_i,t_{i+1}[} : k, l \in [\![1..N]\!]^2 \ \& \ k \neq l$ de cette matrice (termes non diagonaux) correspondent à la valeur de la disponibilité de début de traversée du segment connectant le nœud k au nœud l pendant la période $[t_i, t_{i+1}[$. Les termes $a_{k,k}^{[t_i,t_{i+1}[} : k \in [\![1..N]\!]$ correspondent à la disponibilité du nœud k pendant la période $[t_i, t_{i+1}[$.

4.10.4.2 Calcul du chemin permettant d'arriver au plus tôt

La stratégie de détermination du chemin permettant d'arriver au plus tôt utilise une version modifiée de l'algorithme A* que l'on trouve en Algo. 9. Cette version est adaptée aux graphes d'intervalles de début de traversée. Dans cet algorithme, on peut accéder à l'instance temporelle s_t de l'intervalle $[t_i, t_{i+1}[$ d'un nœud s à un instant $t_a \in [t_i, t_{i+1}[$ par l'intermédiaire de la fontion `getTemporalInstance(`s`,`t_a`)`. L'implication du temps dans le problème opère des changements fondamentaux dans le fonctionnement de l'algorithme, car une instance temporelle de nœud (ITN) peut être quittée à partir du moment où elle est atteinte, et jusqu'à ce que son intervalle de temps correspondant soit révolu. Par conséquent, l'expansion d'une ITN s_t, qui est atteinte à l'instant $t_a \in [t_i, t_{i+1}[$, se fait vers toutes les instances temporelles des nœuds qui sont reliées à s_t pendant $[t_i, t_{i+1}[$ (qui correspondent aux nœuds situés au bout des segments partant de s_t dont la traversée peut commencer entre t_i et t_{i+1}), et qui sont atteignables en partant de s entre les instants t_a et t_{i+1} (exclu).

Une ITN possède plusieurs attributs :

$s_t.id$: La référence s du nœud qui est associé à l'instance temporelle s_t.

$s_t.l$: La limite inférieur (t_i) de l'intervalle qui est associé à s_t.

$s_t.u$: La limite supérieure (t_{i+1}) de l'intervalle qui est associé à s_t.

$s_t.g$: Le coût du chemin arrivant le plus tôt (*i.e.* la durée du meilleur trajet) trouvé jusque-là, commençant au nœud initial au temps $t = 0$ et finissant en s_t.

$s_t.bp$: La référence à la ITN qui est le meilleur parent de s_t (qui se trouve également à une étape en arrière du meilleur trajet trouvé jusque-là se terminant en s_t).

$s_t.h$: La valeur de la variable d'heuristique de s_t, qui est la même pour toutes les instances temporelle de $s_t.id$. Cette valeur correspond à une estimation optimiste de la durée du trajet entre $s_t.id$ et $sEnd$.

Initialisation : L'algorithme commence par initialiser les ITN de chaque nœud de $\mathcal{N}$ pour chaque intervalle de $\mathcal{D}$. L'attribut g de chaque instance est mis à ∞, sauf la première instance temporelle de $sStart$ pour laquelle $g = 0$. Les valeurs des variables d'heuristiques (h) sont calculée à la ligne 3. Enfin, la première instance temporelle associée à $sStart$ est ajoutée à la liste des instances temporelles prometteuses *open* (ligne 9).

Exploration : Ensuite, l'algorithme entre dans une boucle infinie dans laquelle il explore le graphe. A chaque bouclage, l'algorithme *sélectionne* l'ITN qui est la plus prometteuse, et *étend* sa recherche vers les voisins vers lesquels il peut aller pendant l'intervalle de début de traversé correspondant.

Sélection : L'algorithme commence son bouclage par sélectionner l'ITN contenue dans la liste *open* qui est la plus prometteuse (ligne 11). La fonction `pop` trouve l'ITN s_t contenue dans *open* pour laquelle la somme $s_t.g + s_t.h$ est minimale (Ligne lin:popDefFrancais). Cette somme additionne la durée du meilleur trajet trouvé entre $sStart$ et s_t ($s_t.g$) avec l'estimation optimiste de la durée nécessaire pour joindre $s_t.id$ à $sEnd$ ($s_t.h$). Par conséquent, cette somme correspond à une estimation optimiste de la durée du trajet partant de $sStart$ à $t = 0$, et allant à $sEnd$, en passant par

input :

- $\mathcal{G}(\mathcal{N},\mathcal{E},\mathcal{D})$ // a step-in interval graph ($\mathcal{G}$) comprised of nodes ($\mathcal{N}$), of edges ($\mathcal{E}$) and dates set ($\mathcal{D}$).
- sStart // the reference to the starting node
- sEnd // the reference to the destination node

output: path // an ordered list of nodes associated to a timing

```
1  Procedure AStarIntervalGraph(G(N,E,D), sStart, sEnd)
2      forall s ∈ N do
3          heur ← heuristicDistanceBetween(s, sEnd);
4          forall [t_i, t_{i+1}[ ∈ D do
5              createTemporalInstance(s,[t_i, t_{i+1}[,heur,E);
6          end
7      end
8      open ← ∅, closed ← ∅;
9      s_t ← getTemporalInstance(sStart, 0), s_t.g ← 0, add(s_t,open);
10     loop
11         s_t ← pop(open);
12         if s_t.id = sEnd then break loop;
13         else if s = ∅ then return ∅;
14         foreach n ∈ stepInNeighboursOf(s_t,E) do
15             costEdge ← cost(s_t.id, n, E);
16             firstArrival ← s_t.g + costEdge, lastArrival ← s_t.u + costEdge ;
17             t ← firstArrival;
18             while t < lastArrival do
19                 n_t ← getTemporalInstance(n, t) ;
20                 if n_t ∉ closed, t < n_t.g then
21                     n_t.g ← t;
22                     n_t.bp ← s_t;
23                     if n_t ∉ open then add(n_t,open);
24                 end
25                 t ← n_t.u;
26             end
27         end
28         if canStayIn(s_t) then expandToNextTemporalInstance(s_t);
29         remove(s_t,open), add(s_t,closed);
30     end
31     rpath ← ∅, append(s_t, rpath);
32     while s_t.id ≠ sStart do
33         p_t ← s_t.bp;
34         s_t ← p_t;
35         append(s_t, rpath);
36     end
37     path ← reverse(rpath) , return path ;
38 end
```

Algorithm 9: Une variation de l'algorithme A* adapté pour les graphes à intervalles de début de traversée pour lequel la traversée des segment n'est pas instantanée. Les modifications par rapport à Algo. 1 sont montré en bleu foncé.

```
1  Procedure createTemporalInstance(s,[t_i, t_{i+1}[,heur,E)
2      s_t ← ∅, s_t.id ← s, s_t.l ← t_i, s_t.u ← t_{i+1}, s_t.h ← heur, s_t.g ← ∞, s_t.bp ← ∅;
3      linkSandSt(s,s_t);
4  end
5  Procedure pop(open)
6      s_t ← arg min_{c_t ∈ open}(c_t.g + c_t.h);
7      return s_t;
8  end
9  Procedure expandToNextTemporalInstance(s_t)
10     if s_t.u ≠ ∞ then
11         c_t ← getTemporalInstance(s_t.id,c_t.l);
12         c_t.g ← c_t.l;
13         c_t.bp ← s_t;
14         if c_t ∉ open then add(c_t,open);
15     end
16 end
```

Algorithm 10: Implementation des procédures `createTemporalInstance`, `pop` et `expandToNextTemporalInstance`.

$s_t.id$.

Expansion : Ensuite, chaque voisin de s_t dont le segment associée peut commencer à être traversé est identifié à la ligne 14. Les dates de la première arrivée ($firstArrival$) et de la dernière arrivée ($lastArrival$) à ce voisin n, en prenant un trajet commençant à $sStart$ à $t = 0$ et arrivant à n en passant par s_t sont calculées (ligne 16). Puis, toutes les instances temporelles de ce nœud dans l'intervalle $[firstArrival, lastArrival[$, sont ajoutées à *open* si elle ne l'ont pas déjà été. Leur attribut g et leur meilleur parent bp sont mis à jour si le trajet passant par s_t fournit une meilleure option de trajet.

Après avoir fait l'expansion à tous ses voisins, un expansion finale de l'ITN s_t est effectuée vers l'instance temporelle de $s_t.id$ correspondant à l'intervalle suivant (ligne 28). Pour cette instance temporelle, le coût du meilleur chemin devient donc $s_t.u$ (*i.e.*

$s_t.u$ correspond à la borne supérieure de validité de s_t, et aussi à la borne inférieure de l'instance temporelle de l'intervalle suivant), et le meilleur parent devient s_t. Si un autre chemin permettait d'arriver en même temps dans cette instance temporelle, on choisit quand même de mettre à jour le parent pour qu'il devienne s_t. Cette priorisation a un effet sur l'endroit où l'agent choisira de faire ses temporisations quand deux trajets optimums équivalents possèdent le même chemin géométrique et un timing différent. Cette subtilité force l'agent à avancer le plus loin possible dans son chemin géométrique avant de temporiser.

Critères d'arrêt : La boucle s'arrête dès que `pop` ne renvoie aucun nœud (ligne 12) où quand il sélectionne une instance temporelle de *sEnd* (ligne 13). Dans le premier cas, cela signifie que la liste *open* est vide, et qu'il n'existe plus de candidat pour étendre la recherche. Par conséquent, il n'existe aucune solution permettant de lier *sStart* à *sEnd*. Dans le second cas, la sélection d'une instance de *sEnd* implique qu'un trajet reliant *sStart* à *sEnd* peut être retrouvé en remontant de parent en parent à partir de *sEnd*.

Reconstruction du trajet : Quand la condition d'arrêt de la ligne 13 est atteinte, le trajet peut être reconstruit (lignes 31-37). L'instance temporelle de *sEnd* (qui était stockée dans s_t au dernier appel de `pop`) est ajoutée à une liste *rpath* (pour "reversed path"). Ensuite, l'algorithme exploite la chaîne des meilleurs parents, en partant de cette instance temporelle, pour remonter jusqu'à la première instance temporelle de *sStart*. En inversant l'ordre des instances temporelle stockées dans *rpath*, le trajet optimal vis à vis de la date d'arrivée est retrouvé. L'emploi du temps des points de démarrage et d'attente dans les nœuds peut facilement être calculée en comparant les valeurs de g de deux nœuds successifs avec le coût de la traversée du segment les reliant.

Un exemple d'utilisation de cet algorithme est décrit dans le corps principal de la thèse à la Section 4.5.

Bibliography

[1] Robotiq. *Cobots ebook*. Collaborative Robots Buyers Guide, Robotiq.com, 2018.

[2] David Busson, Richard Béarée, and Adel Olabi. Task-oriented rigidity optimization for 7-dofs redundant manipulators. 2017. IFAC WC 2017, Toulouse.

[3] David Busson and Richard Béarée. A pragmatic approach to exploiting full force capacity for serial redundant manipulators. *IEEE Robotics and Automation Letters*, 3(2):888–894, 2018.

[4] J. Denavit and R. S. Hartenberg. A kinematic notation for lower-pair mechanisms based on matrices. *Trans. of the ASME. Journal of Applied Mechanics*, 22:215–221, 1955.

[5] Wisama Khalil and J Kleinfinger. A new geometric notation for open and closed-loop robots. In *Procedings of the IEEE International Conference on Robotics and Automation, 1986.*, volume 3, pages 1174–1179.

[6] Kenneth J. Waldron and James Schmiedeler. *Kinematics*, pages 11–36. Springer International Publishing, Cham, 2016.

[7] Gene H Golub and Christian Reinsch. Singular value decomposition and least squares solutions. *Numerische mathematik*, 14(5):403–420, 1970.

[8] Charles A Klein and Anthony A Maciejewski. Singular value decomposition: computation and applications to robotics. 1989.

[9] Jan Snyman. *Practical mathematical optimization: an introduction to basic optimization theory and classical and new gradient-based algorithms*, volume 97. Springer Science & Business Media, 2005.

[10] Richard M Murray. *A mathematical introduction to robotic manipulation*. CRC press, 2017.

[11] Fedor Menasevich Dimentberg. The screw calculus and its applications in mechanics. Technical report, FOREIGN TECHNOLOGY DIV WRIGHT-PATTERSONAFB OH, 1968.

[12] Jean Lemaitre, Jean-Louis Chaboche, Ahmed Benallal, and Rodrigue Desmorat. *Mécanique des matériaux solides-3eme édition*. Dunod, 2009.

[13] Bruno Siciliano and Oussama Khatib. *Springer handbook of robotics*. Springer, 2016.

[14] Michael E Plesha, Gary L Gray, and Francesco Costanzo. *Engineering Mechanics: Statics*. McGraw-Hill Higher Education, 2010.

[15] Cornelius Lanczos. *The variational principles of mechanics*. Courier Corporation, 2012.

[16] Carl Eckart and Gale Young. The approximation of one matrix by another of lower rank. *Psychometrika*, 1(3):211–218, 1936.

[17] C Radhakrishna Rao. *Generalized inverse of matrices and its applications*. Number 04; QA263, R3. 1971.

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

[19] Alain Liegeois. Automatic supervisory control of the configuration and behaviour of multibody mechanisms. *IEEE Transactions on systems, man and cybernetics*, 7(12):868–871, 1977.

[20] Sarosh Patel and Tarek Sobh. Manipulator performance measures-a comprehensive literature survey. *Journal of Intelligent & Robotic Systems*, 77(3-4):547–570, 2015.

[21] Serdar Kucuk and Zafer Bingul. Comparative study of performance indices for fundamental robot manipulators. *Robotics and Autonomous Systems*, 54(7):567–573, 2006.

[22] Jadran Lenarčič and Carlo Galletti. *Advances in robot kinematics*. Springer, 2000.

[23] Clément M Gosselin. The optimum design of robotic manipulators using dexterity indices. *Robotics and Autonomous systems*, 9(4):213–226, 1992.

[24] Charles A Klein and Bruce E Blaho. Dexterity measures for the design and control of kinematically redundant manipulators. *The International Journal of Robotics Research*, 6(2):72–83, 1987.

[25] Farzam Ranjbaran, Jorge Angeles, and Andrés Kecskeméthy. On the kinematic conditioning of robotic manipulators. In *Procedings of the IEEE International Conference on Robotics and Automation (ICRA), 1996.*, volume 4, pages 3167–3172.

[26] Olav Egeland. Task-space tracking with redundant manipulators. *IEEE Journal on Robotics and Automation*, 3(5):471–475, 1987.

[27] Lorenzo Sciavicco and Bruno Siciliano. A solution algorithm to the inverse kinematic problem for redundant manipulators. *IEEE Journal on Robotics and Automation*, 4(4):403–410, 1988.

[28] Stefano Chiaverini, Giuseppe Oriolo, and Ian D. Walker. *Kinematically Redundant Manipulators*, pages 245–268. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008.

[29] John Baillieul. Kinematic programming alternatives for redundant manipulators. In *Proceedings. 1985 IEEE International Conference on Robotics and Automation.*, volume 2, pages 722–728.

[30] Anthony A Maciejewski and Charles A Klein. Obstacle avoidance for kinematically redundant manipulators in dynamically varying environments. *The international journal of robotics research*, 4(3):109–117, 1985.

[31] Hideo Hanafusa, Tsuneo Yoshikawa, and Yoshihiko Nakamura. Analysis and control of articulated robot arms with redundancy. *IFAC Proceedings Volumes*, 14(2):1927–1932, 1981.

[32] Yoshihiko Nakamura, Hideo Hanafusa, and Tsuneo Yoshikawa. Task-priority based redundancy control of robot manipulators. *The International Journal of Robotics Research*, 6(2):3–15, 1987.

[33] Bruno Siciliano and Jean-Jacques Slotine. A general framework for managing multiple tasks in highly redundant robotic systems. *Advanced Robotics*, pages 1211–1216, 1991.

[34] Joseph Salini, Vincent Padois, and Philippe Bidaud. Synthesis of complex humanoid whole-body behavior: a focus on sequencing and tasks transitions. In *2011 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1283–1290. IEEE, 2011.

[35] Mingxing Liu, Yang Tan, and Vincent Padois. Generalized hierarchical control. *Autonomous Robots*, 40(1):17–31, 2016.

[36] Damir Omrčen, Leon Žlajpah, and Bojan Nemec. Compensation of velocity and/or acceleration joint saturation applied to redundant manipulator. *Robotics and Autonomous Systems*, 55(4):337–344, 2007.

[37] Fabrizio Flacco, Alessandro De Luca, and Oussama Khatib. Motion control of redundant robots under joint constraints: Saturation in the null space. In *IEEE International Conference on Robotics and Automation (ICRA), 2012.*, pages 285–292.

[38] Fabrizio Flacco, Alessandro De Luca, and Oussama Khatib. Prioritized multi-task motion control of redundant robots under hard joint constraints. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2012.*, pages 3970–3977.

[39] Percy Dahm and Frank Joublin. Closed form solution for the inverse kinematics of a redundant robot arm. *Inst. Neuroinf, Ruhr Univ. Bochum*, 44780:97–08, 1997.

[40] Masayuki Shimizu, Hiromu Kakuya, Woo-Keun Yoon, Kosei Kitagaki, and Kazuhiro Kosuge. Analytical inverse kinematic computation for 7-dof redundant manipulators with joint limits and its application to redundancy resolution. *IEEE Transactions on Robotics*, 24(5):1131–1142, 2008.

[41] Yuting Wang. *Closed-form inverse kinematic solution for anthropomorphic motion in redundant robot arms*. PhD thesis, Arizona State University, 2013.

[42] Yin Bu, Wenhe Liao, Wei Tian, Jin Zhang, and Lin Zhang. Stiffness analysis and optimization in robotic drilling application. *Precision Engineering*, 49:388–400, 2017.

[43] Zengxi Pan, Hui Zhang, Zhenqi Zhu, and Jianjun Wang. Chatter analysis of robotic machining process. *Journal of materials processing technology*, 173(3):301–309, 2006.

[44] Bum-Joo Lee. Geometrical derivation of differential kinematics to calibrate model parameters of flexible manipulator. *International Journal of Advanced Robotic Systems*, 10(2):106, 2013.

[45] Yier Wu, Alexandr Klimchik, Anatol Pashkevich, Stéphane Caro, and Benoît Furet. Optimality criteria for measurement poses selection in calibration of robot stiffness parameters. In *ASME 2012 11th Biennial Conference on Engineering Systems Design and Analysis*, pages 185–194. American Society of Mechanical Engineers, 2012.

[46] Claire Dumas, Stéphane Caro, Sébastien Garnier, and Benoît Furet. Joint stiffness identification of six-revolute industrial serial robots. *Robotics and Computer-Integrated Manufacturing*, 27(4):881–888, 2011.

[47] Eberhard Abele, K Schützer, J Bauer, and M Pischan. Tool path adaption based on optical measurement data for milling with industrial robots. *Production Engineering*, 6(4-5):459–465, 2012.

[48] Michael F Zaeh and Oliver Roesch. Improvement of the machining accuracy of milling robots. *Production Engineering*, 8(6):737–744, 2014.

[49] George-Christopher Vosniakos and Elias Matsas. Improving feasibility of robotic milling through robot placement optimisation. *Robotics and Computer-Integrated Manufacturing*, 26(5):517–525, 2010.

[50] Tsuneo Yoshikawa. Dynamic manipulability of robot manipulators. In *Proceedings on the IEEE International Conference on Robotics and Automation (ICRA), 1985*, volume 2, pages 1033–1038. IEEE, 1985.

[51] Yingjie Guo, Huiyue Dong, and Yinglin Ke. Stiffness-oriented posture optimization in robotic machining applications. *Robotics and Computer-Integrated Manufacturing*, 35:69–76, 2015.

[52] P Chiacchio and Mariano Concilio. The dynamic manipulability ellipsoid for redundant manipulators. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA), 1998.*, volume 1, pages 95–100.

[53] E Abele, M Weigold, and S Rothenbücher. Modeling and identification of an industrial robot for machining applications. *CIRP Annals-Manufacturing Technology*, 56(1):387–390, 2007.

[54] Pierre Besset, Adel Olabi, and Olivier Gibaru. Advanced calibration applied to a collaborative robot. In *Power Electronics and Motion Control Conference (PEMC), 2016 IEEE International*, pages 662–667. IEEE, 2016.

[55] Tsuneo Yoshikawa. Manipulability of robotic mechanisms. *The international journal of Robotics Research*, 4(2):3–9, 1985.

[56] Pasquale Chiacchio, Yann Bouffard-Vercelli, and Francois Pierrot. Force polytope and force ellipsoid for redundant manipulators. *Journal of Field Robotics*, 14(8):613–620, 1997.

[57] T Kokkinis and B Paden. Kinetostatic performance limits of cooperating robot manipulators using force-velocity polytopes. In *Proceedings of the ASME Winter Annual Meeting*, pages 151–155, 1989.

[58] Alan Bowling and Oussama Khatib. The dynamic capability equations: a new tool for analyzing robotic manipulator performance. *IEEE transactions on robotics*, 21(1):115–123, 2005.

[59] Bruno Siciliano, Lorenzo Sciavicco, Luigi Villani, and Giuseppe Oriolo. *Robotics: modelling, planning and control*. Springer Science & Business Media, 2010.

[60] P Chiacchio, S Chiaverini, L Sciavicco, and B Siciliano. Influence of gravity on the manipulability ellipsoid for robot arms. *Journal of Dynamic Systems, Measurement, and Control*, 114(4):723–727, 1992.

[61] Pasquale Chiacchio. A new dynamic manipulability ellipsoid for redundant manipulators. *Robotica*, 18(04):381–387, 2000.

[62] Jihong Lee. A study on the manipulability measures for robot manipulators. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 1997.*, volume 3, pages 1458–1465. IEEE, 1997.

[63] O. Khatib and J. F. Le Maitre. Dynamic control of manipulators operating in a complex environment. In *Proc. of the 3rd CISM-IFToMM Symposium on Theory and Practice of Robots and Manipulators*, pages 267–282, Udine, Italy, September 1978.

[64] Dongsheng Zhou, Lu Ji, Qiang Zhang, and Xiaopeng Wei. Practical analytical inverse kinematic approach for 7-dof space manipulators with joint and attitude limits. *Intelligent Service Robotics*, 8(4):215–224, 2015.

[65] Tomás Lozano-Pérez and Michael A Wesley. An algorithm for planning collision-free paths among polyhedral obstacles. *Communications of the ACM*, 22(10):560–570, 1979.

[66] Jacob T. Schwartz and Micha Sharir. A survey of motion planning and related geometric algorithms. In *Artificial Intelligence*. Citeseer, 1988.

[67] Stephen R Lindemann and Steven M LaValle. Current issues in sampling-based motion planning. In *Robotics Research. The Eleventh International Symposium*, pages 36–54. Springer, 2005.

[68] Howie M Choset, Seth Hutchinson, Kevin M Lynch, George Kantor, Wolfram Burgard, Lydia E Kavraki, and Sebastian Thrun. *Principles of robot motion: theory, algorithms, and implementation*. MIT press, 2005.

[69] Jean-Claude Latombe. *Robot motion planning*, volume 124. Springer Science & Business Media, 2012.

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

[71] Héctor H González-Banos, David Hsu, and Jean-Claude Latombe. Motion planning: Recent developments. *Autonomous Mobile Robots: Sensing, Control, Decision-Making and Applications*, 2006.

[72] Nancy M Amato and Guang Song. Using motion planning to study protein folding pathways. *Journal of Computational Biology*, 9(2):149–168, 2002.

[73] Jean-Claude Latombe. Motion planning: A journey of robots, molecules, digital actors, and other artifacts. *The International Journal of Robotics Research*, 18(11):1119–1128, 1999.

[74] Lucas Kovar, Michael Gleicher, and Frédéric Pighin. Motion graphs. In *ACM SIGGRAPH 2008 classes*, page 51. ACM, 2008.

[75] John H Reif. Complexity of the mover's problem and generalizations. In *Foundations of Computer Science, 1979., 20th Annual Symposium on*, pages 421–427. IEEE, 1979.

[76] Shriram Mahabal Udupa. *Collision detection and avoidance in computer controlled manipulators*. PhD thesis, California Institute of Technology, 1977.

[77] Tomas Lozano-Perez. Spatial planning: A configuration space approach. In *Autonomous robot vehicles*, pages 259–271. Springer, 1990.

[78] John Canny. *The complexity of robot motion planning*. MIT press, 1988.

[79] Lydia E. Kavraki and Steven M. LaValle. *Motion Planning*, pages 109–131. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008.

[80] Ellips Masehian and Davoud Sedighizadeh. Classic and heuristic approaches in robot motion planning-a chronological review. *World Academy of Science, Engineering and Technology*, 29(1):101–106, 2007.

[81] Sai Hong Tang, W Khaksar, NB Ismail, and M Ariffin. A review on robot motion planning approaches. *Pertanika Journal of Science and Technology*, 20(1):15–29, 2012.

[82] Mohamed Elbanhawi and Milan Simic. Sampling-based robot motion planning: A review. *Ieee access*, 2:56–77, 2014.

[83] Csaba D Toth, Joseph O'Rourke, and Jacob E Goodman. *Handbook of discrete and computational geometry*. Chapman and Hall/CRC, 2017.

[84] Nils J Nilsson. A mobile automaton: An application of artificial intelligence techniques. Technical report, SRI international, Menlo park CA, artificial intelligence center, 1969.

[85] John Canny. A voronoi method for the piano-movers problem. In *Proceedings of the IEEE International Conference on Robotics and Automation.*, volume 2, pages 530–535.

[86] Daniel Leven and Micha Sharir. Planning a purely translational motion for a convex object in two-dimensional space using generalized voronoi diagrams. *Discrete & Computational Geometry*, 2(1):9–31, 1987.

[87] Osamu Takahashi and Robert J Schilling. Motion planning in a plane using generalized voronoi diagrams. *IEEE Transactions on robotics and automation*, 5(2):143–150, 1989.

[88] Priyadarshi Bhattacharya and Marina L Gavrilova. Roadmap-based path planning-using the voronoi diagram for a clearance-based shortest path. *IEEE Robotics & Automation Magazine*, 15(2), 2008.

[89] Colm Ó'Dúnlaing and Chee K Yap. A "retraction" method for planning the motion of a disc. *Journal of Algorithms*, 6(1):104–111, 1985.

[90] Takao Asano, Tetsuo Asano, Leonidas Guibas, John Hershberger, and Hiroshi Imai. Visibility-polygon search and euclidean shortest paths. In *26th annual symposium on Foundations of Computer Science*, pages 155–164. IEEE, 1985.

[91] Christos Alexopoulos and Paul M Griffin. Path planning for a mobile robot. *IEEE Transactions on systems, man, and cybernetics*, 22(2):318–322, 1992.

[92] Takashi Maekawa, Tetsuya Noda, Shigefumi Tamura, Tomonori Ozaki, and Ken-ichiro Machida. Curvature continuous path generation for autonomous vehicle using b-spline curves. *Computer-Aided Design*, 42(4):350–359, 2010.

[93] Bernard Chazelle. Approximation and decomposition of shapes. *Algorithmic and Geometric Aspects of Robotics*, 1:145–185, 1985.

[94] J Mark Keil and Jorg-R Sack. Minimum decompositions of polygonal objects. In *Machine Intelligence and Pattern Recognition*, volume 2, pages 197–216. Elsevier, 1985.

[95] Rodney A Brooks and Tomas Lozano-Perez. A subdivision algorithm in configuration space for findpath with rotation. *IEEE Transactions on Systems, Man, and Cybernetics*, (2):224–233, 1985.

[96] Mark de Berg, Otfried Cheong, Marc van Kreveld, and Mark Overmars. *Computational geometry: algorithms and applications*. Springer-Verlag TELOS, 2008.

[97] Yoram Koren and Johann Borenstein. Potential field methods and their inherent limitations for mobile robot navigation. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA), 1991.*, pages 1398–1404.

[98] Gianluca Antonelli, Filippo Arrichiello, and Stefano Chiaverini. The null-space-based behavioral control for autonomous robotic systems. *Intelligent Service Robotics*, 1(1):27–39, 2008.

[99] Michael Defoort, Annemarie Kokosy, Thierry Floquet, Wilfrid Perruquetti, and Jorge Palos. Motion planning for cooperative unicycle-type mobile robots with limited sensing ranges: A distributed receding horizon approach. *Robotics and autonomous systems*, 57(11):1094–1106, 2009.

[100] José Mendes Filho, Eric Lucet, and David Filliat. Real-time distributed receding horizon motion planning and control for mobile multi-robot dynamic systems. In *IEEE International Conference on Robotics and Automation (ICRA), 2017*, 2017.

[101] AD Blackowiak and SD Rajan. Multi-path arrival estimates using simulated annealing: application to crosshole tomography experiment. *IEEE Journal of Oceanic Engineering*, 20(3):157–165, 1995.

[102] Qidan Zhu, Yongjie Yan, and Zhuoyi Xing. Robot path planning based on artificial potential field approach with simulated annealing. In *IEEE Sixth International Conference on Intelligent Systems Design and Applications (ISDA), 2006.*, volume 2, pages 622–627.

[103] Min Gyu Park, Jae Hyun Jeon, and Min Cheol Lee. Obstacle avoidance for mobile robots using artificial potential field approach with simulated annealing. In *Industrial Electronics, 2001. Proceedings. ISIE 2001. IEEE International Symposium on*, volume 3, pages 1530–1535. IEEE, 2001.

[104] Vincent Roberge, Mohammed Tarbouchi, and Gilles Labonté. Comparison of parallel genetic algorithm and particle swarm optimization for real-time uav path planning. *IEEE Transactions on Industrial Informatics*, 9(1):132–141, 2013.

[105] Yuan-Qing Qin, De-Bao Sun, Ning Li, and Yi-Gang Cen. Path planning for mobile robot using the particle swarm optimization with mutation operator. In *Proceedings of the International Conference on Machine Learning and Cybernetics, 2004.*, volume 4, pages 2473–2478.

[106] MA Porta Garcia, Oscar Montiel, Oscar Castillo, Roberto Sepúlveda, and Patricia Melin. Path planning for autonomous mobile robot navigation with ant colony optimization and fuzzy cost function evaluation. *Applied Soft Computing*, 9(3):1102–1110, 2009.

[107] Kazuo Sugihara and John Smith. Genetic algorithms for adaptive motion planning of an autonomous mobile robot. In *Computational Intelligence in Robotics and Automation, 1997. CIRA'97., Proceedings., 1997 IEEE International Symposium on*, pages 138–143. IEEE, 1997.

[108] Jianping Tu and Simon X Yang. Genetic algorithm based path planning for a mobile robot. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), 2003.*, volume 1, pages 1221–1226.

[109] AT Ismail, Alaa Sheta, and Mohammed Al-Weshah. A mobile robot path planning using genetic algorithm in static environment. *Journal of Computer Science*, 4(4):341–344, 2008.

[110] Adem Tuncer and Mehmet Yildirim. Dynamic path planning of mobile robots with improved genetic algorithm. *Computers & Electrical Engineering*, 38(6):1564–1572, 2012.

[111] Lynne E Parker. Path planning and motion coordination in multiple mobile robot teams. *Encyclopedia of complexity and system science*, pages 5783–5800, 2009.

[112] Ralph Beckers, OE Holland, and Jean-Louis Deneubourg. From local actions to global tasks: Stigmergy and collective robotics. 1994.

[113] Barry Brian Werger. Cooperation without deliberation: A minimal behavior-based approach to multi-robot teams. *Artificial Intelligence*, 110(2):293–320, 1999.

[114] C Ronald Kube and Hong Zhang. Collective robotics: From social insects to robots. *Adaptive behavior*, 2(2):189–218, 1993.

[115] Iraj Hassanzadeh and Sevil M Sadigh. Path planning for a mobile robot using fuzzy logic controller tuned by ga. In *Mechatronics and its Applications, 2009. ISMA'09. 6th International Symposium on*, pages 1–5. IEEE, 2009.

[116] Xiaoyu Yang, Mehrdad Moallem, and Rajnikant V Patel. A layered goal-oriented fuzzy motion planning strategy for mobile robot navigation. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 35(6):1214–1224, 2005.

[117] Toshio Fukuda and Naoyuki Kubota. An intelligent robotic system based on a fuzzy approach. *Proceedings of the IEEE*, 87(9):1448–1470, 1999.

[118] Qingfu Zhang, Jianyong Sun, Gaoxi Xiao, and Edward Tsang. Evolutionary algorithms refining a heuristic: A hybrid method for shared-path protections in wdm networks under srlg constraints. *IEEE TRANSACTIONS ON SYSTEMS MAN AND CYBERNETICS PART B*, 37(1):51, 2007.

[119] Takanori Shibata and Toshio Fukuda. Intelligent motion planning by genetic algorithm with fuzzy critic. In *Proceedings of the 1993 IEEE International Symposium on Intelligent Control, 1993.*, pages 565–570. IEEE, 1993.

[120] Steven M LaValle. Rapidly-exploring random trees: A new tool for path planning. 1998.

[121] Lydia E Kavraki, Petr Svestka, J-C Latombe, and Mark H Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE transactions on Robotics and Automation*, 12(4):566–580, 1996.

[122] Sertac Karaman and Emilio Frazzoli. Sampling-based algorithms for optimal motion planning. *The international journal of robotics research*, 30(7):846–894, 2011.

[123] Lucas Janson, Edward Schmerling, Ashley Clark, and Marco Pavone. Fast marching tree: A fast marching sampling-based method for optimal motion planning in many dimensions. *The International journal of robotics research*, 34(7):883–921, 2015.

[124] Sean R Martin, Steve E Wright, and John W Sheppard. Offline and online evolutionary bi-directional rrt algorithms for efficient re-planning in dynamic environments. In *IEEE International Conference on Automation Science and Engineering (CASE) 2007.*, pages 1131–1136.

[125] Léonard Jaillet, Juan Cortés, and Thierry Siméon. Transition-based rrt for path planning in continuous cost spaces. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) 2008.*, pages 2145–2150.

[126] Oktay Arslan and Panagiotis Tsiotras. Use of relaxation methods in sampling-based algorithms for optimal motion planning. In *IEEE International Conference on Robotics and Automation (ICRA), 2013*, pages 2421–2428.

[127] Jonathan D Gammell, Siddhartha S Srinivasa, and Timothy D Barfoot. Informed rrt*: Optimal sampling-based path planning focused via direct sampling of an admissible ellipsoidal heuristic. *arXiv preprint arXiv:1404.2334*, 2014.

[128] Michael Otte and Emilio Frazzoli. Rrt x: Real-time motion planning/replanning for environments with unpredictable obstacles. In *Algorithmic Foundations of Robotics XI*, pages 461–478. Springer, 2015.

[129] Edward F Moore. The shortest path through a maze. In *Proc. Int. Symp. Switching Theory, 1959*, pages 285–292, 1959.

[130] Chin Yang Lee. An algorithm for path connections and its applications. *IRE transactions on electronic computers*, (3):346–365, 1961.

[131] Konrad Zuse. *Der Plankalkül*. Number 63. Gesellschaft für Mathematik und Datenverarbeitung, 1972.

[132] Lester R Ford Jr. Network flow theory. Technical report, RAND CORP SANTA MONICA CA, 1956.

[133] Claude Berge. La theorie des graphes. *Paris, France*, 1958.

[134] H Cormen Thomas, E Leiserson Charles, L Rivest Ronald, and Clifford Stein. Section 24.3: Dijkstra's algorithm. *Introduction to Algorithms*, pages 595–601, 2001.

[135] Peter E Hart, Nils J Nilsson, and Bertram Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE transactions on Systems Science and Cybernetics*, 4(2):100–107, 1968.

[136] Lydia E Kavraki and Mihail N Kolountzakisy Jean-Claude Latombe. Analysis of probabilistic roadmaps for path p lanning. *IEEE Transactions on Robotics and Automation*, 14(1):166–171, 1998.

[137] Roland Geraerts and Mark H Overmars. A comparative study of probabilistic roadmap planners. In *Algorithmic Foundations of Robotics V*, pages 43–57. Springer, 2004.

[138] Gildardo Sánchez and Jean-Claude Latombe. A single-query bi-directional probabilistic roadmap planner with lazy collision checking. In *Robotics Research*, pages 403–417. Springer, 2003.

[139] Marcelo Kallman and Maja Mataric. Motion planning using dynamic roadmaps. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA), 2004.*, volume 5, pages 4399–4404. IEEE, 2004.

[140] Peter Leven and Seth Hutchinson. A framework for real-time path planning in changing environments. *The International Journal of Robotics Research*, 21(12):999–1030, 2002.

[141] Tobias Kunz, Ulrich Reiser, Mike Stilman, and Alexander Verl. Real-time path planning for a robot arm in changing environments. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5906–5911.

[142] G. Kurts, G. Lucas (Producers), and I. Kersner (Director). The empire strikes back. *Star Wars, Twentieth Century Fox*, 1980.

[143] Jur Van Den Berg, Dave Ferguson, and James Kuffner. Anytime path planning and replanning in dynamic environments. In *ICRA 2006. Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006.*, pages 2366–2371. IEEE, 2006.

[144] Maxim Likhachev, David I Ferguson, Geoffrey J Gordon, Anthony Stentz, and Sebastian Thrun. Anytime dynamic a*: An anytime, replanning algorithm. In *ICAPS*, pages 262–271, 2005.

[145] B Bui Xuan, Afonso Ferreira, and Aubin Jarry. Computing shortest, fastest, and foremost journeys in dynamic networks. *International Journal of Foundations of Computer Science*, 14(02):267–285, 2003.

[146] Petter Holme and Jari Saramäki. Temporal networks. *Physics reports*, 519(3):97–125, 2012.

[147] William Hedley Thompson and Peter Fransson. The frequency dimension of fmri dynamic connectivity: network connectivity, functional hubs and integration in the resting brain. *NeuroImage*, 121:227–242, 2015.

[148] Prithwish Basu, Amotz Bar-Noy, Ram Ramanathan, and Matthew P Johnson. Modeling and analysis of time-varying graphs. *arXiv preprint arXiv:1012.0260*, 2010.

[149] Lambiotte Renaud and Masuda Naoki. *A Guide To Temporal Networks*, volume 4. World Scientific, 2016.

[150] Alex Pentland and Andrew Liu. Modeling and prediction of human behavior. *Neural computation*, 11(1):229–242, 1999.

[151] Akinori Asahara, Kishiko Maruyama, Akiko Sato, and Kouichi Seto. Pedestrian-movement prediction based on mixed markov-chain model. In *Proceedings of the 19th ACM SIGSPATIAL international conference on advances in geographic information systems*, pages 25–33. ACM, 2011.

[152] Nicolas Schneider and Dariu M Gavrila. Pedestrian path prediction with recursive bayesian filters: A comparative study. In *German Conference on Pattern Recognition*, pages 174–183. Springer, 2013.

[153] Judith Bütepage, Michael J Black, Danica Kragic, and Hedvig Kjellström. Deep representation learning for human motion prediction and classification. In *IEEE*

Conference on Computer Vision and Pattern Recognition (CVPR), page 2017. IEEE, 2017.

[154] Judith Bütepage, Hedvig Kjellström, and Danica Kragic. Anticipating many futures: Online human motion prediction and synthesis for human-robot collaboration. *arXiv preprint arXiv:1702.08212*, 2017.

[155] Yong Du, Wei Wang, and Liang Wang. Hierarchical recurrent neural network for skeleton based action recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1110–1118, 2015.

[156] Fei Han, Brian Reily, William Hoff, and Hao Zhang. Space-time representation of people based on 3d skeletal data: A review. *Computer Vision and Image Understanding*, 158:85–105, 2017.

[157] Katerina Fragkiadaki, Sergey Levine, Panna Felsen, and Jitendra Malik. Recurrent network models for human dynamics. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4346–4354, 2015.

[158] Stéphanie Lefèvre, Dizan Vasquez, and Christian Laugier. A survey on motion prediction and risk assessment for intelligent vehicles. *Robomech Journal*, 1(1):1, 2014.

[159] Allison Bruce and Geoffrey Gordon. Better motion prediction for people-tracking. In *Proc. of the Int. Conf. on Robotics & Automation (ICRA), Barcelona, Spain*, 2004.

[160] Thibault Kruse, Amit Kumar Pandey, Rachid Alami, and Alexandra Kirsch. Human-aware robot navigation: A survey. *Robotics and Autonomous Systems*, 61(12):1726–1743, 2013.

[161] Jorge Rios-Martinez, Anne Spalanzani, and Christian Laugier. From proxemics theory to socially-aware navigation: A survey. *International Journal of Social Robotics*, 7(2):137–153, 2015.

[162] Harmish Khambhaita and Rachid Alami. Viewing robot navigation in human environment as a cooperative activity. *arXiv preprint arXiv:1708.01267*, 2017.

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

[164] Christoph Rösmann, Wendelin Feiten, Thomas Wösch, Frank Hoffmann, and Torsten Bertram. Trajectory modification considering dynamic constraints of autonomous robots. In *7th German Conference on Robotics; Proceedings of ROBOTIK 2012.*, pages 1–6. VDE, 2012.

[165] Wisama Khalil and Etienne Dombre. *Modeling, identification and control of robots*. Butterworth-Heinemann, 2004.

GESTION DE MANIPULATEURS MOBILES ET REDONDANTS EN ENVIRONNEMENT CONTRAINT ET DYNAMIQUE

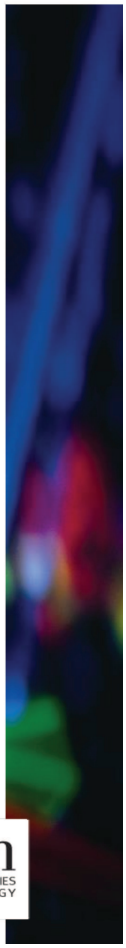
RESUME : L'utilisation de robots collaboratifs dans l'industrie de production est en plein essor. Ces robots, dont la puissance est limitée, sont dotés de capteurs leur permettant de détecter la présence d'obstacles, afin de garantir la sécurité des humains se trouvant aux alentours. On s'intéresse dans cette thèse à l'utilisation de systèmes redondants, collaboratifs et mobiles (bras articulés montés sur plateformes mobiles), dans un environnement de production aéronautique peuplé d'humains, pour la réalisation d'opérations d'assemblage. Du point de vue des process, l'utilisation de ces systèmes, souvent beaucoup moins imposants et rigides que leurs homologues non collaboratifs, est jalonnée de défis. La grande souplesse mécanique et les faibles couples qui les caractérisent peuvent induire des imprécisions de positionnement et une incapacité à soutenir l'intensité d'une interaction physique. Ce contexte induit également un besoin d'autonomie de ces systèmes, qui sont amenés à travailler dans des environnements en perpétuelle évolution. Dans cette thèse, une formulation de la redondance cinématique est d'abord présentée. Le formalisme associé permet de simplifier l'exploitation de la liberté que ces systèmes possèdent sur le choix des postures à utiliser pour réaliser des tâches de placement statique de l'effecteur. Ce formalisme est ensuite exploité pour améliorer et caractériser le comportement en déformation et la capacité d'application d'efforts des systèmes redondants sériels. Enfin, le sujet de la planification des mouvements de systèmes robotisés dans un environnement dynamique et encombré est considéré. La solution présentée adapte l'algorithme bien connu des Probabilistic RoadMaps pour y inclure une anticipation des trajectoires des obstacles dynamiques. Cette solution permet de planifier des mouvements sécuritaires, peu intrusifs et efficaces jusqu'à la destination.

Mots clés : Robotique industrielle, Redondance, Précision robot, Capacité d'effort, Planification de mouvements, Anticipation.

MANAGEMENT OF MOBILE AND REDUNDANT MANIPULATORS IN CLUTTERED AND DYNAMIC ENVIRONMENT

ABSTRACT : Industrial applications involving collaborative robots are regarded with a growing interest. These power-limited systems are embedded with additional sensing capabilities, which allow them to safely work around humans and conquer new industrial grounds. The subject of managing redundant, collaborative and mobile systems, for assembly operations within a human-populated aircraft production environment, is addressed in this thesis. From a process perspective, the use of these smaller and less stiff counterparts of the non-collaborative robots comes with new challenges. Their high mechanical flexibility and weak actuation can cause shortcomings in positioning accuracy or for interaction force sustainment. The ever-changing nature of human-populated environments also requires highly autonomous solutions. In this thesis, a formulation of positional redundancy is presented. It aims at simplifying the exploitation of the freedom redundant manipulators have on static-task-fulfilling postures. The associated formalism is then exploited to characterise and improve the deformational behaviour and the force capacity of redundant serial systems. Finally, the subject of planning motions within cluttered and dynamic environments is addressed. An adaptation of the well-known Probabilistic RoadMaps method is presented – to which obstacles trajectories anticipation has been included. This solution allows to plan safe, efficient and non-intrusive motions to a given destination.

Keywords : Industrial robotics, Redundancy, Robot accuracy, Force capacity, Motion planning, anticipation.



ARTS ET MÉTIERS ParisTech

ParisTech
INSTITUT DES SCIENCES ET TECHNOLOGIES
PARIS INSTITUTE OF TECHNOLOGY