Développement d'un algorithme de cinématique d'interaction appliqué sur un bras robotique dans un contexte de coopération humain-robot

Mémoire

Philippe LeBel

Maîtrise en génie mécanique - avec mémoire
Maître ès sciences (M. Sc.)

Québec, Canada

Développement d'un algorithme de cinématique d'interaction appliqué sur un bras robotique dans un contexte de coopération humain-robot.

Mémoire

Philippe Lebel

Sous la direction de:

Alexandre Campeau-Lecours, directeur de recherche
Clément Gosselin, codirecteur de recherche

Résumé

Ce mémoire présente le développement d'un algorithme de cinématique d'interaction dans un contexte de coopération humain-robot. Cet algorithme vise à faciliter le contrôle de bras robotiques en évitant les collisions, singularités et limites articulaires du robot lorsqu'il est contrôlé par un humain. La démarche adoptée pour l'ateinte de l'objectif est le prototypage d'un algorithme sur une structure robotique simple, la validation expérimentale de ce prototype, la généralisation de l'algorithme sur une base robotique à 6 degrés de liberté et la validation de l'algorithme final en le comparant expérimentalement avec un algorithme similaire.

Premièrement, le prototype d'algorithme est développé sur un bras robotisé Jaco, de l'entreprise Kinova, duquel le poignet a été retiré. Cette architecture permet le déplacement de l'effecteur selon 3 degrés de liberté en translation. L'algorithme développé sur cette base robotique permet :

— l'évitement des collisions avec les objets présents dans l'environnement de travail,

— l'évitement des limitations articulaires

— et l'évitement des singularités propres à l'architecture du robot.

Les performances de l'algorithme sont ensuite validées lors d'expérimentations dans lesquelles il a été démontré que l'algorithme a permis une réduction d'approximativement 50% du temps de complétion d'une tâche donnée tout en réduisant l'attention que l'utilisateur doit porter sur le contrôle du robot comparativement à l'attention portée à l'accomplissement de la tâche demandée.

Ensuite, des améliorations sont apportées à l'infrastructure :

— une méthode de numérisation de l'environnement de travail est ajoutée,

— un meilleur algorithme de détection de collisions et de mesure des distances minimales entre les membrures du robot et les obstacles présents dans l'environnement de travail est implémenté.

De plus, une méthode de balayage de l'environnement de travail à l'aide d'une caméra Kinect ainsi qu'un algorithme de segmentation de nuage de points en polygones convexes sont présentés. Des tests effectués avec l'algorithme prototype ont été effectués et ont révélé que bien que des imperfections au niveau de la méthode de balayage existent, ces modifications de l'infrastructure peuvent améliorer la facilité avec laquelle l'algorithme de cinématique d'interaction peut être implémenté.

Finalement, l'algorithme implémenté sur une architecture robotique à six degrés de liberté est présenté. Les modifications et les adaptations requises pour effectuer la transition avec la version initiale

de l'algorithme sont précisées. Les expérimentations ont validé la performance de l'algorithme vis à vis un autre algorithme de contrôle pour l'évitement de collisions. Elles ont démontré une amélioration de 25% en terme de temps requis pour effectuer une tâche donnée comparé aux temps obtenus avec un algorithme de ressorts-amortisseurs virtuels.

Table des matières

Liste des tableaux

Liste des figures

Remerciements

Merci à mes parents pour m'avoir offert tout leur support durant mes études universitaires. 7 ans d'université, c'est long, mais ça en valait la peine !

Je tiens aussi à remercier mes directeurs de maîtrises, Alexandre Campeau-Lecours et Clément Gosselin, pour m'avoir permis de réaliser ce projet de recherche tout en poursuivant mon implication (quelques fois un peu trop intense) dans mon projet étudiant Robocup Ulaval. Malgré le fait que durant certaines périodes je dédiais une bonne proportion de mon temps d'étude sur ce projet, vous êtes restés ouverts et je vous en suis très reconnaissant.

Finalement, merci à ma conjointe Florence pour m'avoir aidé à surmonter des épreuves rencontrées au cours de ces années d'étude.

Avant-propos

Ce mémoire est présenté sous la forme d'un mémoire par articles. Les articles composant le mémoire sont alors listés avec leur statut de publication, ma contribution dans la rédaction de chacun d'eux et les modifications effectuées pour les besoins du mémoire.

- **Chapitre 1 :**

  Titre : *An Anticipative Kinematic Limitation Avoidance Algorithm For Collaborative Robots : Three-Dimensional Case*

  Type d'article : Article de conférence, IROS 2017.

  Statut : Publié, 14 décembre 2017.

  Contribution : Auteur principal, rédaction de l'entièreté de l'article avec les conseils et avis des coauteurs.

  Coauteurs : Prof. Alexandre Campeau-Lecours et Prof. Clément Gosselin ont supervisé la réalisation et la finalité de cet article.

  Modifications : Article dans sa version originale.

- **Chapitre 2 :**

  Titre : *Améliorations proposées pour un algorithme d'évitement de collisions d'un robot sériel à trois degrés de liberté.*

  Type d'article : Article académique.

  Statut : Non soumis.

  Contribution : Auteur principal, rédaction des sections 2.1, 2.2, 2.4, 2.7 et 2.8.

  Coauteurs : Philippe Babin et Jérôme Landuré ont participé à la rédaction de l'article et à la réalisation des expérimentations.

  Modifications : Article dans sa version originale mis à part la conclusion qui est modifiée pour tenir compte du contexte de l'implémentation des différents algorithmes (environnement de recherche versus industriel).

- **Chapitre 3 :**

  Titre : *A Constraint Based Kinematic Limitation Avoidance : The Sliding Algorithm For Six-Dimensional Collaborative Robots.*

  Type d'article : Article de journal.

Statut : Non soumis. Robotics and automation letters.

Contribution : Auteur principal, rédaction de l'entièreté de l'article avec les conseils et avis des coauteurs.

Coauteurs : Prof. Alexandre Campeau-Lecours a supervisé la réalisation et la finalité de cet article.

Modifications :

— Article présenté avant les modifications pour diminuer sa taille à 8 pages (taille maximale pour les conférences ICRA et IROS) sous le format standard de l'IEEE. L'article fait présentement 8,25 pages de long, alors les modifications requises pour réduire sa taille seront mineures et n'influenceront pas l'information véhiculée par l'article.

— Section additionnelle pour l'exemplification du procédé de Gram Schmidt.

Introduction

Contexte et problématique générale

Les avancées effectuées dans le domaine de la robotique favorisent l'utilisation de solutions robotisées tant en entreprise que dans le domaine de la réadaptation. Des solutions robotiques sont maintenant développées pour résoudre des problématiques allant du travailleur ayant besoin d'un système robotique pour effectuer des tâches ardues et peu ergonomiques jusqu'au cas de la personne ayant des difficultés motrices voulant utiliser un bras robotique pour gagner de l'autonomie. On dénote, dans ces cas d'application, un rapprochement considérable de l'humain et du robot lors de la réalisation des tâches. Cependant, ces cas de figures ne sont pas pris en compte correctement par la plupart des solutions robotisées présentement implémentées en industrie. Le paradigme présentement en place consiste plutôt en la ségrégation des machines robotisées et des humains dans le milieu de travail. Les algorithmes alors utilisés pour le déplacement des robots, tels que des bras robotiques, ne prévoient pas la nécéssité de changements de trajectoire pour éviter des collisions dans l'espace de travail. Les tâches préprogrammées réalisées par ces robots sont sensibles à des variations de l'environnement de travail, tel qu'un nouvel objet placé dans l'environnement ou un humain passant près du robot, ce dernier pourrait ne pas s'adapter en conséquence, ce qui pourrait entrainer des blessures ou des bris de matériel. La stratégie mise en place pour éviter ces conséquences négatives est l'arrêt complet du robot dès qu'un humain s'en approche ou bien qu'une collision survienne. Cependant, pour les groupes d'utilisateurs mentionnés plus haut, il est peu ergonomique, voire même impossible, d'utiliser cette technique de contournement du problème. Puisque les robots et les humains évoluent dans un espace commun, cet espace de travail est voué à être parsemé d'obstacles et la tâche que les robots effectueraient dans cet environnement de travail devrait constamment changer. Il est alors nécessaire de rechercher une solution qui permettera au robot de travailler dans un environnement où des obstacles sont présents, où la trajectoire planifiée pourra être modifiée à tout moment et qui permettra à un humain de contrôler le bras robotique facilement. L'utilisation d'algorithmes permettant de moduler le comportement de bras robotiques face à différentes situations de collisions est la solution préconisée dans plusieurs travaux tels que [2], [3], [4], [5]. Cependant, ces algorithmes ne permettent que le pré-calcul de trajectoires d'évitement et ne sont pas facilement applicables dans le contrôle direct d'un bras robotique par un utilisateur. Puisque l'utilisateur peut avoir de la difficulté à anticiper la position finale du bras robotique pour l'accomplissement d'une tâche donnée, il serait particulièrement

difficile d'adapter ces algorithmes pour résoudre la problématique. C'est pourquoi l'élaboration d'un algorithme qui modifie la commande envoyée par l'utilisateur en fonction des collisions et autres limitations rencontrées par le bras robotique, tout en essayant de rester fidèle à l'intention de la commande initiale, est la méthode à utiliser dans de tels cas.

C'est pour l'atteinte de cet objectif que le projet de développement d'un algorithme de cinématique d'interaction pour la collaboration humain-robot a été mis sur pied.

Objectifs de recherche

L'objectif du projet de recherche est de développer un algorithme de contrôle permettant à un utilisateur, n'ayant pas ou peu d'expérience avec la manipulation de bras robotiques, d'accomplir une tâche donnée sans entrer en collision avec des objets présents dans l'environnement de travail et sans avoir à se soucier des limitations des joints du robot et des singularités présentes dans l'environnement de travail du robot.

De manière plus spécifique, l'algorithme devra traiter les commandes de vitesses envoyées au robot par un utilisateur et valider si cette commande engendrera une collision avec un objet présent dans l'environnement de travail ou si elle rapprochera le robot d'une limite articulaire ou d'une singularité. Si tel est le cas, l'algorithme devra modifier la commande de l'utilisateur en "glissant" sur les limitations identifiées. Ces caractéristiques permettront ultimement de contrôler avec aisance un bras robotique sans entrainement préalable pour l'utilisateur et sans connaissances des notions de singularités et des limitaions articulaires du robot.

Afin de faciliter le développement de cet algorithme, le projet est divisé en deux volets :

1. Développement d'une version préliminaire de l'algorithme permettant l'accomplissement des objectifs lorsque appliqué sur une architecture robotique simple ayant 3 degrés de liberté (DDL) en translation (x, y, z).
2. Développement de la version finale de l'algorithme permettant l'accomplissement des objectifs lorsque appliquée sur une architecture robotique standard ayant 6 degrés de liberté, 3 en translation et 3 en rotation.

Méthodologie et plan du mémoire

La méthodologie utilisée pour chacun des volets est la suivante :

1. Sélection de la base robotique sur laquelle l'algorithme sera implémenté.
2. Élaboration de l'infrastructure logicielle permettant de communiquer avec le robot et d'obtenir une lecture des orientations des joints.
3. Conception d'un simulateur permettant de visualiser la base robotique dans un environnement virtuel.

4. Détermination de la cinématique directe et inverse de la base robotique.
5. Développement de l'algorithme avec l'aide du simulateur.
6. Validation de la performance de l'algorithme par comparaison de performance avec d'autres algorithmes d'évitement de collisions lors d'expérimentations avec l'aide de participants n'ayant pas ou peu d'expérience en robotique.

Tout d'abord, l'algorithme développé pour résoudre la problématique du premier volet du projet de recherche est présenté. C'est dans le Chapitre 1 que toute l'infrastructure algorithmique requise pour le bon fonctionnement de la solution est détaillée. Il est décrit :

– le développement d'un algorithme implémenté de maniére pratique avec des techniques qui permetteraientt sa mise en application dans une situation réelle,
– l'élaboration d'un algorithme spécialisé dans la détection de collision,
– le scan 3D de l'environnement de travail est effectué permettant de bien démontrer les performances de l'algorithme dans un milieu de travail.

L'élaboration de cette mise en action de l'algorithme est présentée au Chapitre 2.

Finalement, l'algorithme doit être généralisé à un système robotisé à 6 DDL tel que défini par le deuxième volet du projet de recherche. C'est au Chapitre 3 que le résultat final du projet de recherche est alors présenté.

Chapitre 1

An Anticipative Kinematic Limitation Avoidance Algorithm For Collaborative Robots : Three-Dimensional Case

Résumé

Cet article présente un algorithme anticipatif d'évitement de limitations cinématiques pour les robots collaboratifs. L'objectif de ce projet est de simplifier le contrôle d'un bras robotique en vue d'améliorer les performances des utilisateurs dans un contexte de coopération humain-robot. Présentement, dans de telles interactions, l'utilisateur doit détourner une grande partie de ça concentration vers le contrôle du bras robotique plutôt que de se concentrer totalement sur la tâche pour laquelle il utilise le robot. La raison de ce détournement d'attention se résume au fait que le contrôle d'un bras robotique est parsemé d'embûches telles que les limitations articulaires, les singularités et les collisions potentielles avec des objets ou des humains présents dans l'environnement de travail. L'approche utilisée pour diminuer la complexité de contrôle est le développement d'un algorithme qui évite automatiquement les problèmes mentionnés plus haut et qui tente de respecter les commandes envoyées par l'utilisateur. Premièrement, l'écosystème développé pour la gestion de plusieurs limitations simultanées dans un environnement à 3 dimensions est présenté. Ensuite, l'algorithme est détaillé pour chaque cas de limitations pouvant survenir lors de l'utilisation d'un robot sériel à 3 degrés de liberté en translation de type RRR. Finalement, des résultats d'expérimentation sont présentés pour permettre d'évaluer la performance de l'algorithme développé.

Abstract

This paper presents an anticipative robot kinematic limitation avoidance algorithm for collaborative robots. The main objective is to improve the performance and the intuitiveness of physical human-robot interaction. Currently, in such interactions, the human user must focus on the task as well as on the robot configuration. Indeed, the user must pay a close attention to the robot in order to avoid limitations such as joint position limitations, singularities and collisions with the

environment. The proposed anticipative algorithm aims at relieving the human user from having to deal with such limitations by automatically avoiding them while considering the user's intentions. The framework developed to manage several limitations occurring simultaneously in three-dimensional space is first presented. The algorithm is then presented and detailed for each individual limitation of a spatial RRR serial robot. Finally, experiments are performed in order to assess the performance of the algorithm.

1.1 Introduction

Collaborative robots working alongside humans are now used in many fields such as industrial applications, service robotics and medical applications [6]. By harnessing the strengths of humans and robots simultaneously, human-robot interaction has the potential to increase human performance and to provide effective assistance in many applications. However, in order to achieve this, the interaction between the human user and the robotic device must be safe and intuitive [6], [7], [8]. To this end, the robot motion can be designed to behave similarly to that of human beings [9], [10]. Additionally to being intuitive and predictable, the robot motion should also be legible, that is to enable the collaborator to quickly and confidently infer the robot's goal. This aspect was introduced and detailed in [8]. While collaborative robots allow a close cooperation with humans, commercial devices still automatically stop when a limitation (joint position limitation, singularities and collision with the environment) is reached, which is clearly not intuitive. In order to avoid the limitations, the user would have to pay a close attention to said limitations, which can be very difficult and cognitively demanding (especially for internal kinematic limitations such as position limitations and singularities).

The management of limitations such as self-collisions and collisions with the environment has extensively be explored in the literature [2], [3], [4], [5]. However, many of the proposed trajectory planning algorithms are designed for industrial robots where the trajectory is planned off-line. Collaborative robots bring new challenges that may not be effectively handled by the existing techniques. For instance, collaborative robots are usually used in unstructured and dynamic environments and must manage many real-time constraints such as collocation with humans [11]. Additionally, while existing trajectory planning algorithms are effective at finding an optimal path between two points, in a human-robot collaboration application the final destination is often unknown by the planner. Indeed, the user can directly control the robot's direction and velocity (through a joystick or by direct manipulation of the robot) and can thus bring the robot in prohibited configurations. The management of the robot limitations must then be as transparent as possible in order to prevent the user from having to constantly monitor these limitations, which would be detrimental to the principal task.

In order to solve this problem, many solutions have been proposed in the literature such as potential fields [12], [13], [14], virtual spring-damper systems [11], [15] and virtual-fixtures [16]. The main advantage of these algorithms is that they are able to repel the robot from the limitations regardless of whether the final destination is known or unknown. These reactive algorithms have been implemented in many applications. For instance, a skeleton algorithm able to manage self-collisions of two 7-DOF

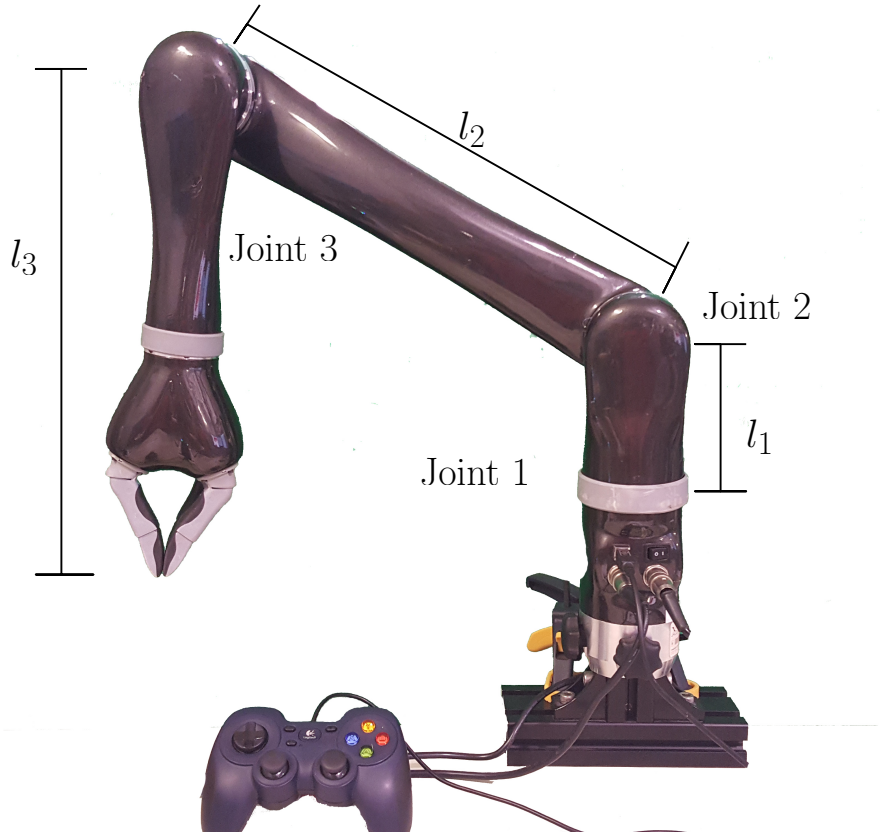

FIGURE 1.1 – 3 DOF modifed Kinova robot arm used for the tests

serial arms in a torso configuration was introduced in [15]. In the proposed scheme, the distance between the robot links is first evaluated and, if necessary, a virtual repulsive force is applied on the joints, by using a virtual spring-damper system, in order to avoid self-collisions.

Even if these methods are successful at avoiding collisions, they present inherent drawbacks. Their reactive nature produces a change of path only after the trespassing of a limitation. To avoid a collision with an object, the buffer zone around the object must either be much larger than the actual object or present a high stiffness that guarantees that the robot cannot penetrate too far in the limitation. These two methods of managing the reactive nature of the algorithms require an extensive experimental tuning. The former method needs to validate if the space between the limitation boundary and the object is large enough to avoid a collision. The latter needs to verify that the large stiffness of the repulsive force does not produce vibrations or instability [17]. Even after tuning these parameters, the reaction of the robot is still hard to predict for a human user since the virtual force applied on the links varies with the velocity and acceleration of the robot when it collides with an object. These algorithms also have trouble dealing with multiple limitations. Some limitations may contradict each other, pushing the robot in the direction of a less restricting limitation.

The algorithm presented in this paper addresses these considerations by anticipating limitations ins-

tead of reacting to them. This anticipative limitation avoidance algorithm aims at making the robot control efficient, intuitive and safe by transforming the user's input in an easily predictable way. The algorithm analyzes the limitations near the robot and computes the components of the user's input that may make the robot collide with a limitation. These components are then removed thus allowing the robot to slide alongside multiple simultaneous limitations with multiple points of contact.

This paper is structured as follows. A brief description of the algorithm's structure and of the virtual limitation detection are presented. Then, the sliding algorithm for a single point defined on the effector is presented. The algorithm is then expanded to the case of multiple and simultaneous contacts. Experimental results are then reported in order to assess the performance of the algorithm. Finally, the results are discussed and a conclusion is drawn.

1.2 General structure of the algorithm

The framework required to implement the sliding algorithm includes the following components :

- Limitations defined in the robot's workspace.
- Proximity detection algorithm.
- Angular position of the robot's joint.
- A velocity verification algorithm.

The limitations can be defined as external or internal. External limitations include obstacles in the workspace as well as protection zones defined by users. Internal limitations consist of the angular limits of the joints, singularities and self-collision of robot links.

Fig.1.2. provides an overview of the algorithm. The velocity requested by the user, $\vec{v}_u$, is fed to the Jacobian matrix in order to determine the requested joint velocity $\dot{\theta}_r$. It is also used by the limitation detection algorithm in order to determine the constraints to be used by the sliding algorithm. The algorithm is then applied to modify the requested user velocity $\vec{v}_u$ and produce the effective Cartesian velocity $\vec{v}_m$, which is converted into the joint velocity array that is finally sent to the robot.

1.3 Proximity detection algorithm

In order to avoid objects in close proximity of the robot, a collision and distance detection algorithm is used. Its purpose is to compute normal vectors pointing toward the closest point of interest on the robot.

Different approaches can be used to compute the shortest distance between clouds of points or planes that may compose the robot model and the limitations. Algorithms such as the GJK algorithm [18] and the octree [19] are refined techniques used in video games that take advantage of geometric relations to reduce the required number of points to process.

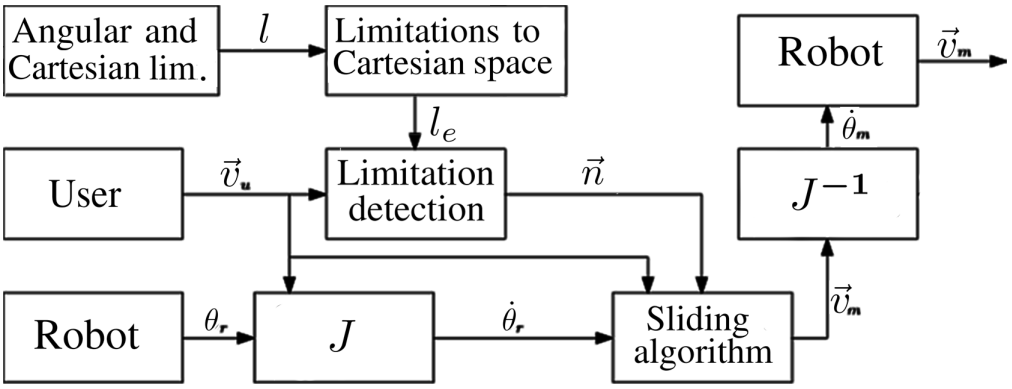

FIGURE 1.2 – General structure of the algorithm.

The normal vectors extracted from these algorithms are then used as limitations by the sliding algorithm.

1.4 Limitation sliding algorithm for a single point

This section presents the proposed algorithm that allows a single point defined along one of the robot's links, such as the end-effector's reference point, to slide on limitations. Since the robot evolves in a three-dimensional space, only three non-redundant limitations are required to fully constrain and immobilize the robot. A larger number of limitations is redundant and can be reduced to a set of three nonlinearly dependent limitations. Accordingly, it is only possible to slide on a maximum of two limitations simultaneously. When computing the end-effector's modified velocity, all the limitations must be expressed in the same set of coordinates in order to process them simultaneously. If the limitations are not processed simultaneously, for instance if some limitations are processed in the articular space and others in the Cartesian space, contradictory sliding results may occur [9].

The method proposed to process the limitations simultaneously can be described as follows :

At first, the limitations are transformed into Cartesian limitations of the point of interest on the robot. Then, the active limitations are identified. For a limitation to be considered active, the scalar product of the end-effector velocity vector requested by the user and the limitation's normal vector must be negative. An allowed direction of the requested Cartesian velocity is determined for every combination of two limitations. The only possible direction that allows the sliding on the two limitations simultaneously is the direction defined by the line corresponding to the intersection of the two limitation planes. This direction is obtained using the cross product of the two normal vectors of the limitation planes. The user's input velocity is then projected on each of the allowed directions vectors. The projections are then tested to determine if they satisfy all the limitations. The projections that fail the test are discarded. The same user's velocity is projected on each of the limitation planes. These vectors are also tested and those that fail to satisfy all limitations are discarded in the same manner as in the

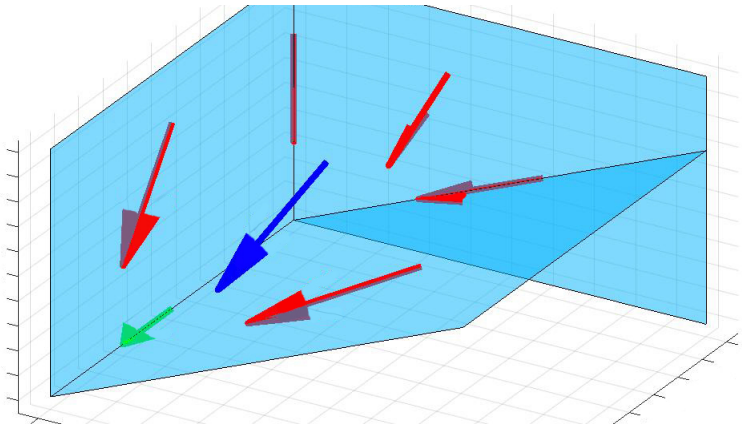

FIGURE 1.3 – Example of application of the proposed algorithm, considering limitations simultaneously. The user's input is displayed in blue and is going toward the first two limitation planes. The only possible direction that satisfies the two limitations and is a component of the user's input is the green vector going along the direction of the line defined by the intersection of the two active limitation planes.

previous step. Finally, the output velocity vector is generated by taking the sum of all the remaining vectors. After this process, only viable directions are remaining and the robot can slide along these directions. Figure 1.3 illustrates this process.

The user's input is displayed as the blue arrow. The green arrow represents the valid directions in which a point can slide on limitations, the red ones represent the components of the user's input that will make the point collide with the planes. In this case, if the limitations were considered independently, the resulting vector would be dependent on which limitation is considered first, sometimes not sliding at all, sometimes trespassing limitations. While a sequential processing of the limitations may work in some simple cases, it renders the algorithm's behaviour difficult to predict and unreliable. It is therefore very important to consider all the limitations simultaneously, which is a contribution of the proposed algorithm.

1.5 Limitation sliding algorithm for Multiple points

In the preceding section, a sliding algorithm was proposed for a single point on the robot. It is also important to consider other points on the robot, such as the elbow as well as other types of limitations, such as angular limitations of the joints and singularities. In order to handle all these limitations simultaneously, the approach proposed here consists in transforming each of the limitation constraints into equivalent constraints at the end-effector. Indeed, it is very important to consider all constraints simultaneously as explained in the preceding section.

1.5.1 Physical limitations and protection zones for multiple, simultaneous, potential collision points

Consider first the reference point on the end-effector of the robot and its velocity vector, $\vec{v}$. The mapping between the joint velocity array, $\dot{\theta}$ and vector $\vec{v}$ is given by the Jacobian matrix of the robot, J, defined at the end-effector, as follows :

$$\vec{v} = J\dot{\theta}. \tag{1.1}$$

Similarly, the relation between the movement of a given point, P, on the robot and the articular velocity array, $\dot{\theta}$, is defined as follows :

$$\vec{v}_p = J_p\dot{\theta} \tag{1.2}$$

where $\vec{v}_p$ is the Cartesian velocity of a given point, P, on the robot, J_p is the corresponding Jacobian matrix (defined at point P) and $\dot{\theta}$ is the joint velocity array of the robot.

When a robot's link approaches a limitation, the collision detection algorithm computes the proscribed direction to avoid the violation of this limit. The only step required before modifying the user's input is to transform this link's forbidden direction into the corresponding end-effector's forbidden direction. This allows the algorithm to process all limitations simultaneously as if they were all limitations of the end-effector motion. Assuming that the robot is not in a singular configuration (matrix J is invertible), eq.(1.1) can be inverted and substituted into eq.(1.2), which yields

$$\vec{v}_p = J_p J^{-1}\vec{v} \tag{1.3}$$

where $\vec{v}_p$ is the velocity of the point at which the collision potentially occurs and J_p is the Jacobian matrix defined at this given point, as described above. This relation allows the computation of the velocity vector of the potential contact point, P, that corresponds to a given velocity, $\vec{\mathbf{v}}$ of the end-effector.

For example, if point P is defined at the elbow of the manipulator of Fig. 1.1 [9], eq.(1.3) allows the determination of the velocity of this point for a given end-effector velocity, $\vec{v}$.

It is however required to perform the inverse transformation in order to find relations between the limitation at the elbow and the corresponding limitation at the end-effector. The result will allow to emulate the limitation by giving a normal vector in the end-effector coordinate system, as if the limitation were encountered at the effector, thus allowing the algorithm to process all the limitations simultaneously.

However, to achieve this result, it is important to take some considerations into account :

1. Given a potential collision point P defined on link i of a d-dof robot, matrix J_p includes some zero columns since the last $(n-i)$ joint velocities have no impact on the velocity of point P.
2. Given the above observation, matrix $\mathbf{J}_p$ is not of full rank and eq.(1.3) cannot be inverted. In the following, the equation that must be used to transfer the limitation at point P into a limitation at the end-effector is presented and the result is an important contribution of this paper.

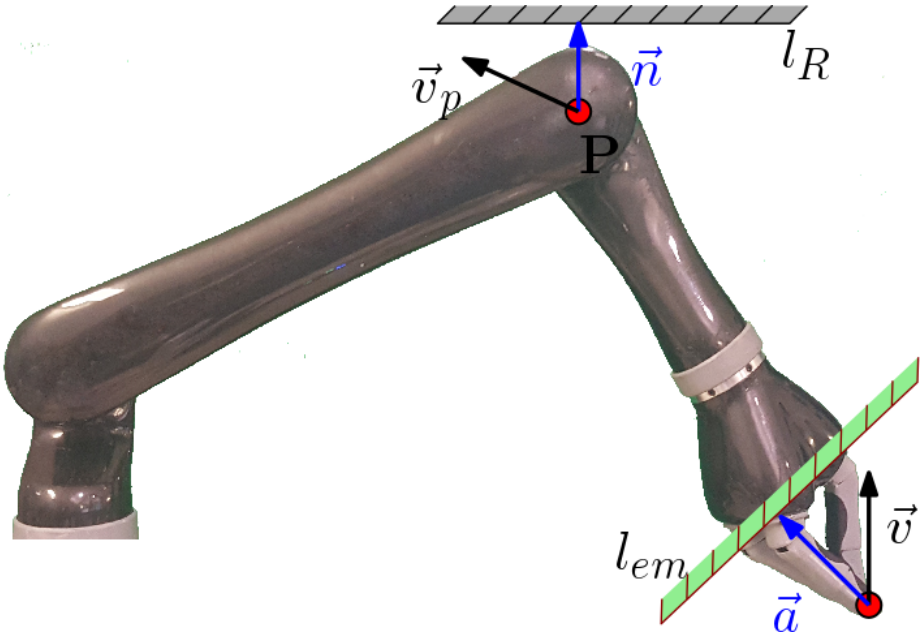

FIGURE 1.4 – Example of transformation of a limitation at point P (defined here at the elbow) into a limitation at the end-effector. Vector $\vec{v}_p$ represents the velocity of the elbow when the velocity $\vec{v}$ is commanded at the end-effector. Vector $\vec{a}$ represents the limitation at the end-effector that corresponds to the limitation at the elbow.

At point P, the constraint associated with an obstacle of normal vector $\vec{n}$ is written as :

$$\vec{n}^T \vec{v}_p = 0. \tag{1.4}$$

Substituting eq.(1.3) into eq.(1.4) then yields

$$\vec{n}^T J_p J^{-1} \vec{v} = 0. \tag{1.5}$$

This equation represents the constraint at point P and is linear in terms of $\vec{v}$. It can be simply written as

$$\vec{a}^T \vec{v} = 0, \tag{1.6}$$

where

$$\vec{a}^T = \vec{n}^T J_p J^{-1}. \tag{1.7}$$

It can be noted that, as the scalar product of $\vec{a}^T$ and $\vec{v}$ is null, vector $\vec{a}$ represents the direction in which the end-effector's velocity must be constrained in order to satisfy the limitation at point P defined on the robot.

Finally, the equation providing the limitation $\vec{a}$ at the end-effector corresponding to a limitation $\vec{n}$ at point P is written as

$$\vec{a} = (J_p J^{-1})^T \vec{n}. \tag{1.8}$$

The limitation $\vec{a}$ can be treated as a limitation occurring at the end-effector and processed simulteanously with the other limitations. An example is shown in Fig. 5, where point P is defined at the elbow. The limitation l_R of normal vector $\vec{n}$ at the elbow is transformed into a limitation l_{em} of normal $\vec{a}$ at the end-effector.

1.5.2 Angular limitation

In order to manage all the limitations simultaneously, the articular limitations must also be transferred into Cartesian limitations at the end-effector. Such a limitation is active when an angular limitation is reached and when the desired motion requires the joint to move in the limitation's direction. In order to find the equations required to transfer the angular limitation constraints into end-effector constraints, the limited joint is considered blocked and the corresponding column of the Jacobian matrix , **J**, defined in eq. (1.1) is replaced with a column of zeros.

Referring to eq.(1.1), it is clear that all the possible Cartesian velocities that satisfy the constraints imposed by the locked joint must be linear combinations of the two remaining columns of the Jacobian matrix J. In other words, this limitation can be expressed as a plane formed by the remaining columns of the 3×3 Jacobian matrix of the robot. The forbidden direction is then computed using the cross product of the two column vectors.

$$\vec{n}_f = \vec{c}_i \times \vec{c}_j \tag{1.9}$$

where c_i and c_j are the non-zero column vectors of matrix **J**. This forbidden direction $\vec{n}_f$ can now be processed simulteanously with all the other limitations.

1.5.3 Singularities

Singularities can be expressed by Cartesian or articular constraints. In both cases, singularity limitations can be expressed in end-effector limitations using the approach presented in Sections 1.5.1 and 1.5.2.

1.6 Experiments

This section presents the experimental protocol along with the results obtained. A comparison with a virtual spring-damper system follows. To assess the effectiveness and intuitiveness of the algorithm, a given task was realized by several participants, both with and without the use of the algorithm. In the latter case, the robot velocity was set to zero when a limitation was reached (as with a standard commercial robot). Subjects were not told which algorithm was used and the order was varied between subjects. The experiments were performed on a modified version of the JACO arm from Kinova shown in Fig. 1.1. Thirteen (13) subjects aged between 21 and 30 participated in the experiments which were approved by the ethics committee of Université Laval certificate no. 2016-311/24-11-2016. The accompanying video shows excerpts from the experiments.

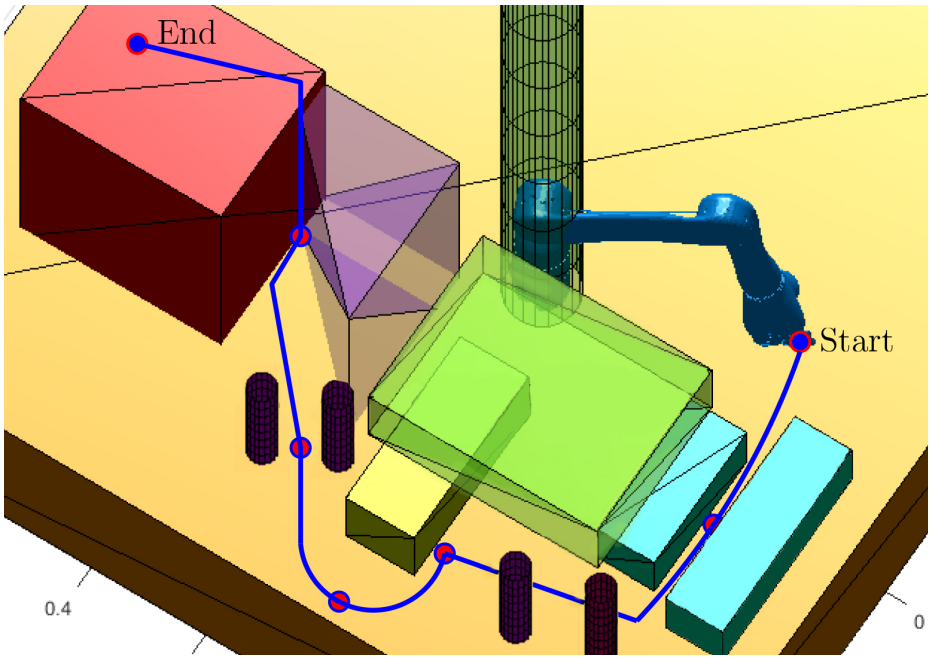

FIGURE 1.5 – Geometric description of the experimental set-up. The polyhedra are physical limitations. The four small cylinders are unprotected obstacles that users must avoid during the task. The blue circled red dots are the way points users must cross. The blue line is an example of a path to complete the task.

1.6.1 The task

The task has been designed to test all cases of possible limitations. Figure 1.5 shows the workspace and the task trajectory.

1.6.2 First experiment : Full attention

In the first experiment, the subjects had to bring the end-effector to the way points shown in Figure 1.5 while avoiding objects in the workspace and the number of collisions with the objects were recorded. Without the sliding algorithm, the mean completion time is 86.5s with a standard deviation of 32.75s and an average of 1.1 collisions. With the sliding algorithm, the mean completion time is 46.4s with a standard deviation of 15.2s and an average of 0.4 collisions.

The mean difference between the completion of the task with and without the algorithm is 87% which corresponds to a difference of 40 seconds. The time difference is considered significant according to a Mann-Whitney-Wilcoxon non parametric test one-tailed ($p= 9.87 * 10^{-5} < 0.05$). It is important to note that the performance improvement is highly dependent on the chosen task. For instance, a task where no limitation is reached would not improve whether the sliding algorithm is used or not. The authors tried to be fair in the selection of the task.

1.6.3 Second experiment : Divided attention

A secondary task is added to the experiment in order to divide the subject's attention (similarly to an industrial task). This secondary task consists in naming a color appearing on a screen at every two seconds. To increase the task's difficulty, the color appearing on the screen is embedded in the font of letters spelling a different color name. For instance, the word blue is spelled on the screen using a red font. The correct answer is then red. The time to complete the task and the total number of errors were recorded. Omitting to name a color is counted as two mistakes, mentioning the wrong color is compiled as one mistake and colliding with an object is counted as three mistakes. This kind of multitasking experiment may help to represent a normal industrial task where external factors could distract the operator or where the operator must pay attention to several elements. In the context of this experiment, the hypothesis is that the sliding algorithm reduces the level of attention required to complete the task. The user would not have to pay attention to the robot's limitations and self-collisions and could thus give more attention to the main task.

Without the sliding algorithm, the mean completion time is 95.3s with a standard deviation of 34.09s and an average of 9.1 mistakes. With the sliding algorithm, the mean completion time is 48.3s with a standard deviation of 9.92s and an average of 1.9 collisions. The mean difference between the completion of the task with and without the algorithm is 97% which corresponds to a difference of 47 seconds. The time difference is considered significant according to a Mann-Whitney-Wilcoxon non parametric test one-tailed ($p= 8.05 * 10^{-6}<0.05$).

1.6.4 Objective evaluation

The participants generally declared that the task was easier to perform with the algorithm. The participants also mentioned that the completion of the task was much more stressful and required much more attention when the algorithm was not activated.

1.6.5 Comparison with other algorithms

As previously mentioned, the proposed sliding algorithm aims to improve performances compared to algorithms found in the litterature. In this section, the proposed sliding algorithm is compared with the reactive virtual spring-damper algorithm [11], [15]. Two main advantages of the proposed sliding algorithm concern the tuning of the parameters and the stability, as detailed in the following.

With the proposed sliding algorithm, the only parameter to adjust is the distance from which the sliding algorithm begins to act. If this minimal distance is too small, the robot may pass through a forbidden limit. This phenomenon is called *clipping* and is often seen in video games and other 3D simulation environments. This distance can also be automatically computed as a function of the robot velocity and thus no parameter tuning is required with the proposed method, which is an important advantage.

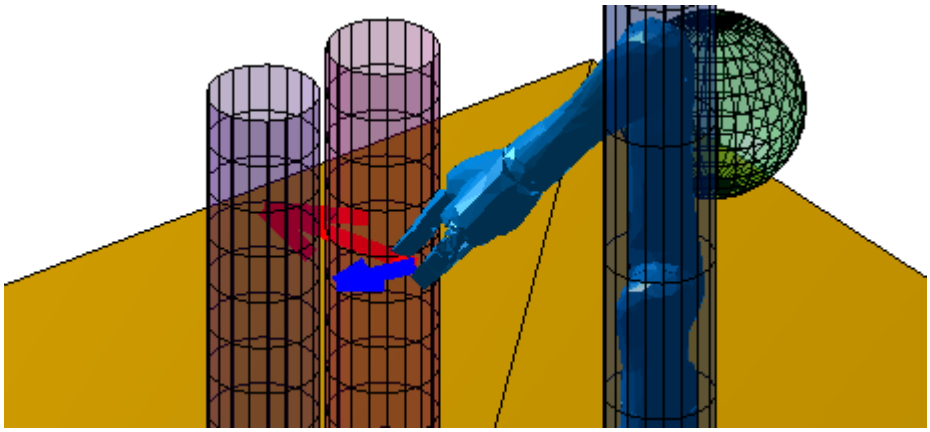

FIGURE 1.6 – Simulation environment. The objective is to pass between the cylinders while avoiding the sphere at the elbow joint.

However, in the virtual spring-damper system, three parameters need to be adjusted, namely the simulated inertia of the end-effector, the stiffness of the virtual spring and the damping coefficient.

The issues that arise from the tuning of these parameters are the following :

- If the stiffness is too low or the simulated mass is too high, the robot may pass through a forbidden surface.
- If the stiffness or damping are too high, the resulting output velocities may not well represent the user's intention.
- Contact with limitations may result in oscillatory behaviour. The occurrence of multiple collisions could result in highly unstable behaviour and the violation of limitations. This phenomenon is even more important when different parts of the robot are in contact with different limitations.
- When going through narrow corridors, the robot might get stuck if its initial velocity is too low.

Figure 1.6 shows an example in which the above last two issues arise. Figure 1.7 shows the corresponding angular position of joint 2 obtained with each of the algorithms.

With the virtual spring-damper method (orange curve), an oscillatory behaviour is observed when the robot begins to interact with the cylinders. The robot also slows down (due to the repulsion) as if a friction force were acting on the end-effector. It is also possible to notice that the robot is unable to continue through the cylinders as it gets stuck between them. With the proposed sliding algorithm (shown in blue) the robot slides along the limitation without being slowed down and no vibrations are observed. This is due to the fact that the algorithm uses an anticipative analytic solution rather than a reactive solution.

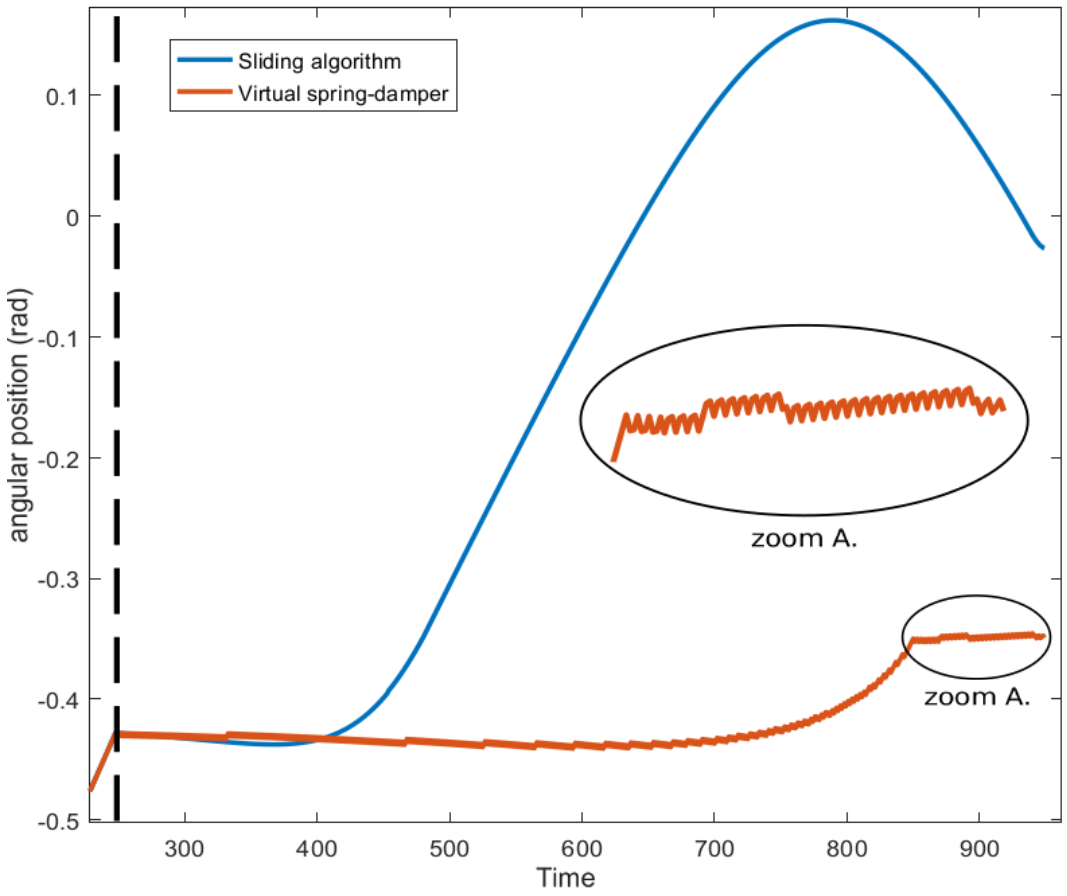

FIGURE 1.7 – Angular positions of the elbow joint (joint 2). The joint position in blue is produced when using the sliding algorithm while the orange curve shows the response using the virtual spring-damper algorithm. The dashed line represents the moment where the robot comes in contact the cylinder.

1.6.6 Real-world implementation

The experiments in this paper were performed with virtual obstacles in order to assess the algorithm's performances. However, in a real dynamic world implementation, the obstacles are detected in real-time by a computer vision system. In order to confirm that the proposed sliding algorithm is compatible with such data, a 3D scanning device (a Microsoft Kinect camera) was used to capture a typical work space as presented in Figure 1.8. The resulting 3D points cloud is segmented by algorithms such as the ones presented in [20] and [21]. Once the segmentation is realized, convex hulls are created to surround the objects.

1.7 Conclusion

This paper presented an anticipative robot kinematic limitation avoidance algorithm for a spatial 3-

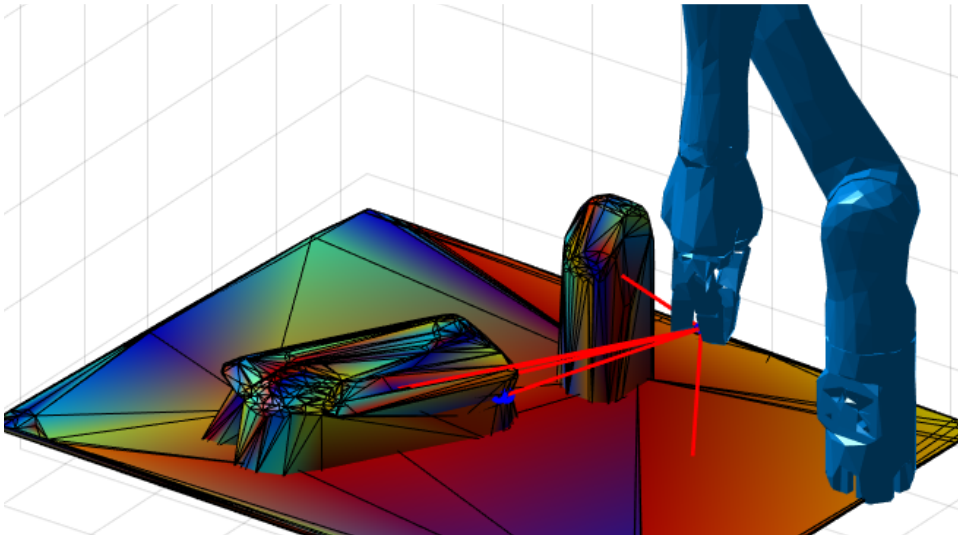

FIGURE 1.8 – 3D scanned environment represented in the simulation space.

DOF serial robot. This algorithm aims at making the robot control easier, more intuitive and safer. Experiments have been performed to observe the efficiency gains while completing tasks performed with the help of the algorithm. One experiment was realized with full attention, another with divided attention. Both tests showed results indicating an increased efficiency (87% and 97% faster completion time) and precision (0.7 and 7.2 fewer errors made) when using the algorithm. These results indicate that the proposed sliding algorithm allows an easier completion of a given task as well as making it safer to accomplish for the robot and the environment. The users are also able to give attention to other secondary tasks while using a serial link manipulator without prior training or knowledge about the basics of the robot's kinematics. Lastly, the algorithm has been compared to other algorithms to confirm that the proposed solution is effective in solving their inherent issues. Further work will be undertaken to generalize the collision detection algorithm to even more complex shapes as well as being able to handle potential collisions with complete surfaces composing the robot arm. 3D cameras could also be utilized to detect objects that are introduced or moved in the workspace during the completion of the task. The algorithm will also be expanded to the 6-DOF case.

Chapitre 2

Améliorations proposées pour un algorithme d'évitement de collisions d'un robot sériel à trois degrés de liberté.

Résumé

Les algorithmes utilisés visent à automatiser la numérisation de l'espace de travail où opère un robot sériel. Une revue de littérature concernant les méthodes utilisées ainsi que la méthodologie de résolution de la problématique seront présentées. L'explication de l'algorithme de détection de collisions suit. La méthode de numérisation de l'environnement est ensuite expliquée. Une description de l'algorithme de segmentation et d'approximation des coques convexes est présentée. Finalement, les résultats d'implémentation sont présentés.

2.1 Introduction

La robotique d'interaction est un domaine qui est de plus en plus présent dans les applications industrielles. L'utilisation des capacités d'adaptation d'un humain combinées à l'endurance et la force d'un système robotique peut paraitre comme étant le summum de l'efficacité en industrie. Cependant, un défi reste à relever : assurer la sécurité de l'opérateur ainsi que minimiser les chances d'accidents ou de bris du matériel adjacent au robot. Pour ce faire, l'implémentation d'un algorithme interprétant l'intention d'un opérateur et essayant de reproduire son intention le plus fidèlement possible tout en évitant les collisions et les singularités a été réalisée comme présenté dans [22].

L'algorithme permet d'éviter les collisions avec des objets programmés dans l'environnement de travail en redirigeant l'effecteur d'un bras robotique lorsqu'une membrure du robot se dirige vers une limitation. Le robot étant contrôlé en temps réel par un utilisateur muni d'une manette, il est impossible de calculer à l'avance une trajectoire pour éviter les collisions. C'est pourquoi l'algorithme présenté dans [22] doit vérifier, en temps réel, la présence de collision entre le robot présenté à la figure 2.1 et chacun des objets présents dans l'environnement. Dans un contexte industriel, l'implémenta-

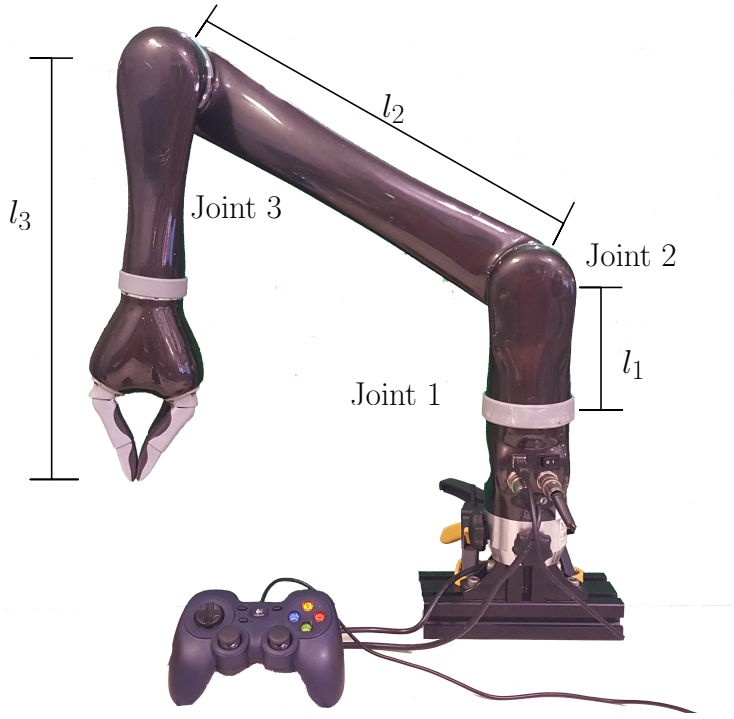

FIGURE 2.1 – Robot sériel 3 degrés de liberté de Kinova utilisé pour les tests

tion d'un algorithme d'évitement de collision capable de prendre en compte le contrôle en temps réel d'un robot permet l'utilisation simplifiée d'un bras robotique. Ainsi, l'utilisateur n'aura pas besoin de comprendre la cinématique du système ni de porter attention à chacune des membrures du robot. Il a aussi été démontré que l'utilisation d'un robot sur lequel l'algorithme a été implémenté rend son utilisation beaucoup plus efficace et réduit le taux d'erreur commises. Jusqu'ici, l'implémentation de l'algorithme nécessitait la programmation et le positionnement manuel précis de chacun des objets dans l'environnement de travail. De plus, l'algorithme de détection de collisions présent ne permettait l'utilisation que de certains types d'objets convexes ; soit les prismes à section convexe et constante, les tubes et les sphères. L'algorithme de détection, présenté dans ce rapport et visant à remplacer celui déjà existant, permet l'utilisation de toute la famille des polygones convexes pour générer les objets dans l'environnement de travail du robot. Ces objets peuvent alors être créés par une méthode de numérisation par une caméra Kinect suivie d'un traitement par un algorithme de segmentation et d'approximation de coques convexes.

Dans un premier temps, une brève revue de littérature concernant les méthodes utilisées ainsi que la méthodologie de résolution de la problématique seront présentées. L'algorithme de détection de collisions sera ensuite présenté. La méthode de numérisation de l'environnement sera alors expliquée. Une description de l'algorithme de segmentation et d'approximation des coques convexes suivra. Finalement, les résultats de mise en pratique seront présentés.

2.2 Description du projet et de la problématique, en lien avec la littérature

La problématique ici est d'implémenter une méthode permettant d'automatiser et simplifier la création de l'environnement de travail requis pour l'implémentation de la solution présentée dans l'article [22].

2.2.1 Littérature liée aux algorithmes de détection de collisions

L'algorithme de détection de collision déjà implémenté s'apparente beaucoup à l'article présenté dans [23]. Cet algorithme, qui s'appuie exclusivement sur les propriétés des prismes à section constante, n'est pas viable dans l'optique de l'implémentation d'une numérisation de l'espace 3D car cette méthode risque de générer des polygones convexes qui ne sont pas à section constante.

Plusieurs algorithmes permettent de calculer la distance minimale entre différents polygones convexes [1]. Une revue de littérature permit d'identifier que l'algorithme le plus performant était l'algorithme de Gilbert-Johnson-Keerthi (GJK) [24].

2.3 Approche proposée

La méthodologie suivante est suivie pour mener à terme le projet :

- l'algorithme de détection de collisions sera implémenté en simulateur,
- la numérisation de l'environnement de travail du robot est effectuée,
- l'algorithme de segmentation et d'approximation en formes convexe est implémenté,
- les deux algorithmes sont ensuite substitué dans le simulateur de bras robotisé développé en [22],
- la mise en oeuvre de la suite d'algorithmes sélectionnés sur le bras robotique est effectuée,
- l'appréciation générale de la performance est cumulée et résumée.

2.4 Algorithme de détection de collision GJK

L'algorithme de détection GJK permet la détection de collisions et le calcul de la distance minimale entre deux polygones convexes composés d'un nombre arbitraire de points. Il est généralisable à des formes convexes inscrites dans un domaine $\mathbb{R}^n$. Cependant, la nature des problématiques nécessitant un algorithme de détection de collision fait en sorte qu'il est très rarement utilisé dans un domaine plus grand que $\mathbb{R}^3$.

Cet algorithme utilise certaines propriétés des solides convexes pour accélérer la convergence tout en n'ayant pas besoin d'examiner chacune des combinaisons de points des deux solides. Il est d'ailleurs démontré dans [24] que l'algorithme peut être implémenté avec une complexité d'ordre $O(a+b)$ où a est le nombre de points constituant un solide et b est le nombre de points constituant l'autre solide.

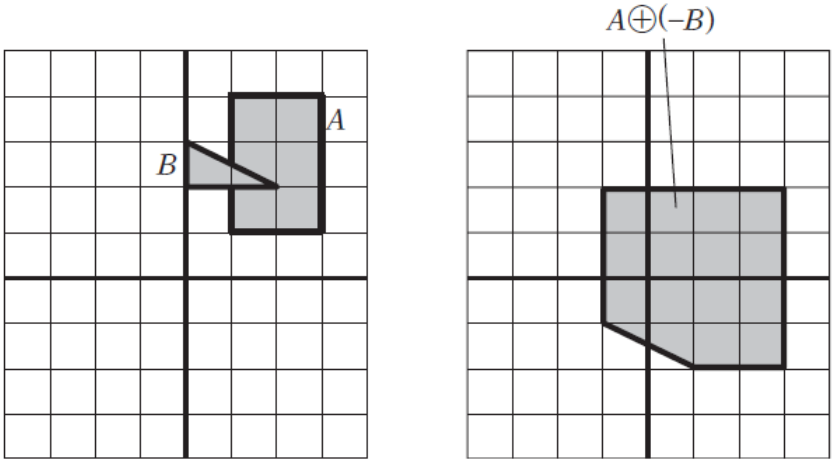

FIGURE 2.2 – La différence de Minkowski est opérée sur les objets A et B. Puisque ces derniers se chevauchent, la différence de Minkowski contient l'origine. (source : [1])

La compréhension de trois éléments clés de l'algorithme est requise pour permettre son implémentation :

- la différence de Minkowski,
- l'utilisation de la méthode du Simplexe dans cette différence de Minkowski,
- le lien entre les coordonnées barycentriques de l'origine et des trois points du *Simplex* et les points de proximité des solides examinés.

2.4.1 Différence de Minkowski

La différence de Minkowski est une opération mathématique opérant sur toutes les combinaisons de points appartenant à deux polygones. N'étant pas mathématiquement définie, elle est décrite comme étant la somme de Minkowski entre le premier objet et la réflexion du deuxième objet par rapport à l'origine. L'équation suivante d'écrit l'opération mathématique réalisée sur les solides A et B [1] :

$$A \ominus B = \{a - b | a \in A, b \in B\}. \tag{2.1}$$

Cette opération crée, en quelque sorte, une combinaison des deux objets. Une propriété intéressante de cette opération mathématique est que la distance séparant la différence de Minkowski et l'origine se traduit directement en la distance séparant les deux solides initiaux. Si la différence de Minkowski se trouve à contenir l'origine, les deux solides initiaux sont nécessairement en collision. La représentation graphique de la différence de Minkowski opérée sur des objets dans le domaine $\mathbb{R}^2$ est représenté dans la figure 2.2.

Il est possible d'observer que la distance de pénétration peut aussi être déduite en trouvant la distance minimale entre une des parois de la différence de Minkowski et l'origine. Pour le reste du rapport, la différence de Minkowski générée par les objets A et B est appelée C.

2.4.2 Recherche de la solution par la méthode du Simplex

L'algorithme GJK utilise une méthode consistant en la construction d'une solution temporaire appelée *Simplex* tout au long des calculs. Le *Simplex* est construit en recherchant des points améliorant la solution d'une itération à l'autre. Dans le cas du calcul des solides inscrits dans un domaine $\mathbb{R}^3$, le *Simplex* sera composé de 3 points constituant un triangle.

L'objectif de l'algorithme est de trouver le triangle, inscrit dans la différence de Minkowski C, qui est le plus près de l'origine. Pour effectuer la recherche des points constituant le *Simplex*, une fonction $support(objet1, objet2, d)$ est utilisée. Cette fonction calcule implicitement la différence de Minkowski et trouve le point compris dans C le plus éloigné dans une direction $\vec{d}$ par rapport à l'origine. Lorsque le *Simplex* commence sa recherche de solution, un point a est choisi aléatoirement. Ensuite, la direction de recherche est définie comme étant le vecteur situé au point a et pointant vers l'origine. Suivant la sélection du deuxième point b, la direction de recherche est définie telle que le vecteur normal à la droite reliant les points a et b maximisant le produit scalaire entre la droite reliant le point a (ou le point b) à l'origine. Le vecteur résultant est tout simplement le vecteur normal à la droite qui pointe dans la direction générale de l'origine.

Une fois le premier *Simplex* construit, la direction de recherche est simplement définie comme étant le vecteur normal au triangle formé par le *Simplex* pointant dans la direction générale de l'origine.

L'évolution de cette solution peu être décrite avec le pseudo code suivant :

Algorithm 1 Algorithme de mise a jour du *Simplex*

```
1:  a = point arbitraire dans C
2:  b = support(objet_1, objet_2, −a)
3:  c = support(objet_1, objet_2, d)
4:  Simplex = [a, b, c]
5:  while True do
6:      SimplexOld = Simplex
7:      d = normale pointant vers l'origine
8:      newPoint = support(objet_1, objet_2, d)
9:      Interchanger newPoint avec le point dans Simplex qui est le plus loin de l'origine.
10:     if SimplexOld == Simplex then
11:         Break
```

2.4.3 Les coordonnées Barycentriques : Le lien entre le domaine de Minkowski et le domaine réel

Maintenant que le *Simplex* a trouvé le triplet de points contenus dans C qui est le plus près de l'origine, il faut maintenant trouver les deux points membres des solides A et B qui sont décrit par les trois points de C. Il faut aussi se rappeler que chacun des points formant C est une combinaison de deux points appartenant à A et B. Tel qu'énoncé dans [1], il est possible d'obtenir les deux points où survient la distance minimale en utilisant les coordonnées barycentriques de l'origine lorsque projetée sur le plan du triangle formé par le *Simplex* inscrit dans C. Ici, les coordonnées barycentriques seront notées c_i, les points membres du *Simplex*, Q_i. Chacun des points de A et B ayant formé le point c_i est alors notés a_i et b_i. La chaine d'équations permettant de trouver les points sur les solides est alors :

$$P = \sum_{i=1}^{3} c_i Q_i, \tag{2.2}$$

où P est le point exact où survient la distance minimale entre la différence de Minkowski C et l'origine. Il est ensuite possible de déduire les points de proximité sur A et B en réalisant que

$$P = \sum_{i=1}^{3} c_i (a_i - b_i) = G - H \tag{2.3}$$

et que par conséquent :

$$G = \sum_{i=1}^{3} c_i a_i, \quad H = \sum_{i=1}^{3} c_i b_i. \tag{2.4}$$

Les points G et H seront alors les points sur les solides A et B, respectivement, où survient la distance minimale. Le calcul de distance est alors trivial.

2.4.4 Considération d'implémentation

Il est à noter qu'implémenter l'algorithme GJK est beaucoup plus ardu que le pseudo-code présenté ci-haut ne peut le laisser paraître. Différents facteurs doivent être pris en considération nottament :

- La dégénération du *Simplex* en une solution contenant des doublons.
- L'oscillation de la solution du *Simplex* entre un nombre aléatoire de solutions (dont une seule est optimale) dans certains cas spécifiques.
- L'identification d'une situation où survient un des deux cas ci-haut et la programmation d'un mécanisme permettant de relancer la convergence.

Dégénérescence du Simplex

La stratégie pour empêcher la dégénérescence du *Simplex* est l'utilisation de critères légèrement différents pour la détermination de la direction de recherche et la substitution du nouveau point avec un des points du *Simplex*. Advenant le cas d'une dégénération du *Simplex*, seuls les deux points formant la ligne la plus rapprochée de l'origine sont gardés. Ensuite, le *Simplex* est reconstruit en suivant la

même logique ayant servi à déterminer la sélection du point c lorsque le *Simplex* n'était formé que de deux points.

Le pseudo-code suivant explique la logique de sélection. Il est à noter que cette partie n'est en aucun cas mentionnée dans l'article original de GJK et est purement le fruit de notre réflexion personnelle sur la résolution du problème.

Algorithm 2 Logique de sélection advenant la dégénération du *Simplex*

$a =$ point arbitraire dans C
$b = support(objet_1, objet_2, -a)$
$c = support(objet_1, objet_2, d)$
$Simplex = [a, b, c]$
while True **do**
    $SimplexOld = Simplex$
    $d =$ normale pointant vers l'origine
    $newPoint = support(objet_1, objet_2, d)$
    if $ismember(newPoint, Simplex)$ **then** le *Simplex* va dégénérer si on ajoute le nouveau point
        $[e, f] =$points du *Simplex* qui forment la ligne du triangle la plus près de l'origine.
        d = Vecteur normal à la droite $\bar{ef}$ pointant dans la direction de l'origine.
        $c = support(objet_1, objet_2, d)$
        $Simplex = [e, f, c]$
    else
        Interchanger *newPoint* avec le point dans *Simplex* qui est le plus loin de l'origine.
    if $SimplexOld == Simplex$ **then**
        Break

Oscillation de la solution

L'oscillation de la solution est résolue par observation. De façon générale l'algorithme GJK converge très rapidement (moins de 4 itérations). Lorsqu'il oscille autour d'une solution, ces oscillations comprennent au plus 10 solutions dont une seule est la solution de distance minimale. Il est alors simple de garder la solution minimale en mémoire pour chaque itération et sortir de la boucle après 10 itérations. Ensuite, une comparaison entre la dernière solution et la solution minimale gardée en mémoire permettent de vérifier si la dernière solution trouvée est bel et bien celle qui identifie la distance minimale. De façon générale, lorsque l'algorithme oscille, c'est parce que plusieurs points sont regroupés près de la solution optimale. Plusieurs triangles peuvent alors être créés et le *Simplex* peine à se stabiliser.

Les solutions proposées parviennent à éliminer une très grande proportion des erreurs. Cependant, l'algorithme implémenté dans la cadre du projet n'est pas parfait, certains de ces cas peuvent parfois faire en sorte que l'algorithme détecte une distance de proximité sensiblement plus grande que ce qu'elle est en réalité. Dans une implémentation en jeu vidéo, de tels évènements seraient bénins, mais dans le contexte du projet, ceci entraine un comportement non voulu du robot. Les impacts seront détaillés dans la section subséquente portant sur l'implémentation sur le robot.

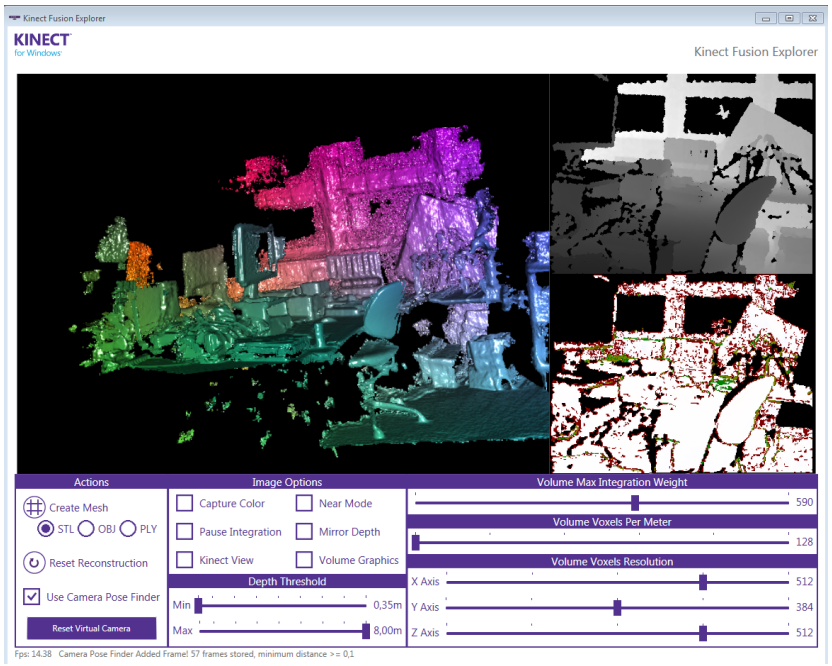

FIGURE 2.3 – Interface de l'application Kinect Fusion Explorer.

2.5 Numérisation de l'environnement

Le système de détection des collisions fonctionne en deux temps. Tout d'abord l'environnement de travail du robot est numérisé pour obtenir une image virtuelle des objets présents dans cet espace de travail, puis le robot est contrôlé en même temps que l'algorithme de détection des collisions qui utilise l'image virtuelle obtenue précédemment pour modifier au besoin la commande de l'utilisateur de manière à éviter les collisions avec les objets de l'espace de travail.

Dans cette section on aborde la méthode utilisée pour faire la numérisation de l'environnement du robot. Pour des raisons pratiques, un 'kinect sensor for Windows' a été choisi. Cet appareil est la première génération de capteur du type 'Kinect' commercialisé par Microsoft. Il intègre une caméra RGB, un émetteur de patron lumineux et un récepteur pour concevoir une caméra qui mesure la profondeur ainsi qu'un microphone. Beaucoup d'outils ont été développés autour du Kinect, notamment une 'trousse à outils de développement' proposé par Microsoft qui intègre des exemples d'applications utilisant le Kinect avec leurs codes sources pour faciliter la prise en main du dispositif. Dans la dernière version en ligne de cette trousse (la V1.8) on trouve notamment des applications de reconstruction 3D intéressantes. Parmi elles, nous nous sommes plus particulièrement intéressés à 'Kinect Fusion Explorer' dont l'interface est visible à la figure 2.3. Cette dernière combine les informations de la 'depth-map camera' à un système de suivi (tracking) pour permettre de recréer en 3D les objets vus par la caméra en faisant se déplacer la caméra autour des objets que l'on souhaite scanner. L'application intègre la possibilité d'exporter le résultat d'une numérisation sous plusieurs formats.

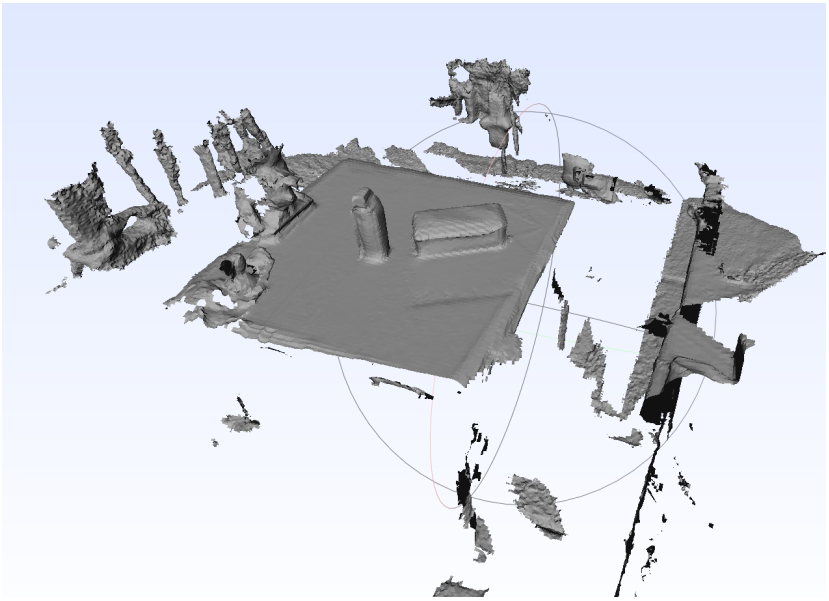

FIGURE 2.4 – Résultat brut obtenu par Kinect Fusion.

2.6 Segmentation de l'environnement

L'étape précédente fourni un objet unique maillée qui représente une image virtuelle de l'environnement. Cet objet n'est pas utilisable en l'état pour plusieurs raisons. Tout d'abord Kinect Fusion va très certainement détecter des zones en dehors de l'espace de travail. Pour les tests à suivre, nous avons constitué l'espace de travail du robot d'une boite et d'une bouteille posées sur une table. La figure 2.4 montre un exemple de numérisation de cet environnement obtenu par Kinect Fusion. Pour enlever tous les éléments superflus, on utilise Meshlab avec lequel on supprime les sommets et les faces inutiles. On obtient alors le résultat de la figure 2.5.

Le maillage obtenu n'est toujours pas exploitable par l'algorithme de détection de collision, car il n'est ni convexe ni même fermé. Les algorithmes de détection de collision nécessitent en général des objets uniquement convexes pour fonctionner correctement du fait de leur nature même. Ce problème est récurrent dans les domaines des simulations numériques et le domaine du jeu vidéo notamment. La solution commune employée quand on est en présence d'un objet non-convexe est de le subdiviser (ou segmenter) en un ensemble de parties convexes. De nombreuses méthodes ont été présentées ces 10 dernières années pour accomplir cette tâche de façon automatisée le plus rapidement et efficacement possible. Par exemple [20] ou [21] sont des méthodes publiées récemment. Pour ce projet nous utilisons WCSeg, car elle est appropriée à notre problème et un support Matlab complet est fourni par les auteurs de cette méthode.
WCSeg est une méthode de segmentation d'objets 3D qui vise à obtenir une découpe sémantique de l'objet, c'est-à-dire que l'objet est découpé en sous-éléments suivant une logique humaine. Par

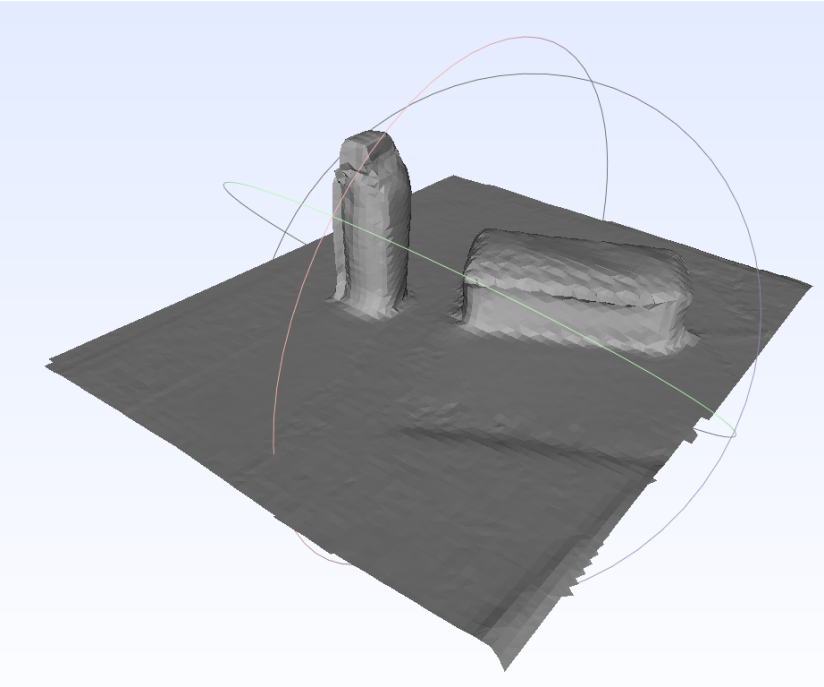

FIGURE 2.5 – Résultat obtenu par Kinect Fusion après nettoyage des éléments superflus.

exemple une tasse se retrouve découpée en son anse et son bol, ces deux éléments n'étant pas convexes.

Cependant WCSeg utilise dans son processus un algorithme de segmentation convexe dite "faible" qui est très suffisant dans notre cas. Ce dernier se base sur la notion de "ligne de vue" qui est une façon de définir un objet convexe. Prenons deux points de l'objet étudié, si la droite qui relie ces deux points reste à l'intérieur de l'objet alors ils ne se voient pas et l'objet reste potentiellement convexe. L'objet est complètement convexe si toutes les combinaisons de deux points parmi les points qui le composent respectent cette règle. Dans le cas de la segmentation convexe faible, on tolère un certain nombre de couples de points qui ne respectent pas cette règle, ce qui évite une subdivision en de trop nombreux éléments. La tolérance sur la convexité est traduite par un coefficient ou critère qui devra être sous une valeur seuil arbitraire. Trop d'éléments nuisent inutilement au temps d'exécution de l'algorithme de détection des collisions qui doit alors gérer beaucoup d'objets en même temps. Le résultat de cette segmentation est visible à la figure 2.6.

Le résultat de cette segmentation est correct. Certaines zones concaves ont été tolérées par l'algorithme comme la zone au niveau du bouchon en haut de la bouteille, la boite a été découpée en plusieurs éléments notamment à cause d'imprécisions de la numérisation de la Kinect qui a créé une forme concave là où il n'y en avait pas. Le nombre total d'éléments crées est 7, on en voit 4 : la bouteille, la table et 3 pour la boite. Les 2 restants sont coincés dans la boite et donc inutiles, ils sont dus à des points parasites qui étaient compliqués à enlever lors du nettoyage et sont donc restés.
Actuellement nous avons subdivisé notre numérisation de l'environnement en 7 éléments à peu près

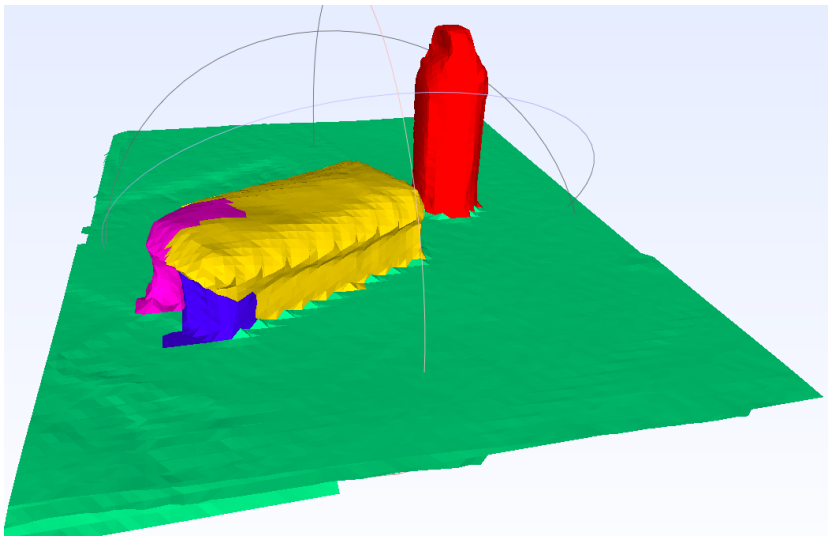

FIGURE 2.6 – Espace de travail segmenté par la méthode WCSeg.

convexes. Ce n'est pas suffisant pour garantir le bon fonctionnement de l'algorithme de détection des collisions, mais on est proche du résultat. Pour finir on utilise une opération qui crée une coque convexe autour d'un objet donné, les objets actuels étant presque convexes ils vont être peu modifiés tout en étant entièrement convexes. L'autre avantage de la création d'une coque convexe est qu'elle réduit fortement le nombre de faces et sommets de l'objet. La numérisation passe de 23510 faces et 71756 sommets à 1408 faces et 718 sommets. Le résultat final est visible à la figure 2.7.

2.7 Montage Expérimental

Pour valider les méthodes décrites dans les sections précédentes, un exemple d'application a été monté. Un robot d'assistance personnelle JACO vendu par l'entreprise KINOVA était disponible pour nos essais et a été choisi pour l'expérience. Ce robot est un bras à 6 degrés de liberté maximum dont l'usage premier est de permettre à des personnes à mobilité réduite de gagner en autonomie. Pour simplifier l'intégration pour l'expérimentation seuls les 3 degrés de liberté en translation sont utilisés. L'environnement choisi pour l'essai est celui dont la numérisation par le Kinect est visible sur la figure 2.4. C'est une table sur laquelle sont posées une boite de mouchoirs et une bouteille de produit d'entretien. Le robot est placé de sorte que son espace de travail coïncide le plus possible avec l'environnement numérisé. Ensuite, la numérisation est traitée de la façon décrite dans le chapitre sur la segmentation, puis l'algorithme de gestion des collisions GJK décrit dans la section qui lui est dédiée a été implémenté dans le contrôle du JACO. Le résultat de ce test est visible dans la vidéo en annexe de ce document. Dans un premier temps, on voit le JACO faire son initialisation de position de façon automatique, puis le contrôle passe aux mains d'un utilisateur par le biais d'une manette de jeu paramétrée pour gérer les translations du robot. La stratégie de commande empêche les collisions en retirant la partie du vecteur de déplacement demandé par l'utilisateur (flèche bleue sur la simulation

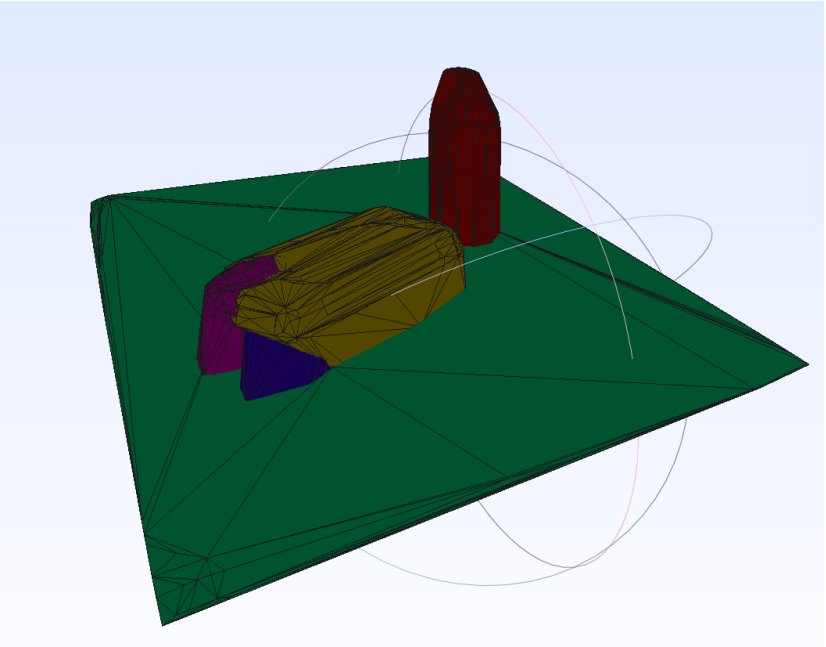

FIGURE 2.7 – Espace de travail final segmenté en éléments convexes.

dans la vidéo) qui entrainerait une collision quand la position estimée de l'effecteur est proche des objets virtuels dans le simulateur, objets images des objets dans l'espace de travail du robot. Il en résulte le vecteur de commande de déplacement rouge. En pratique, quand l'effecteur est proche des objets à éviter et qu'il demande un déplacement qui entrainerait une collision, alors l'effecteur donne l'impression de glisser sur les contours de l'objet. Ensuite on voit le robot décrire les contours de la boite, puis la bouteille. Enfin une trajectoire de contour complète est effectuée.

2.8 Conclusion

L'objectif était de développer un protocole de contrôle d'un robot qui éviterait les collisions avec les objets présents dans son environnement. Une méthode qui fusionne la capacité de numérisation 3D du Kinect, l'algorithme de segmentation WCSeg, l'algorithme de détection des collisions GJK et une implémentation avec un robot JACO a été présentée dans ce document. La méthode, bien qu'imparfaite, fonctionne bien. Il faut simplement s'assurer de faire une numérisation complète par le Kinect, qui peut s'avérer ardue car des points non existants peuvent apparaitre, faussant ainsi la détection virtuelle des collisions.

Pour des raisons pratiques reliées aux ressources de traitement disponibles, un ajustement de la vraisemblance de la numérisation avec la réalité a été nécessaire pour le traitement de la numérisation et de la segmentation convexe. Les temps de calcul ont donc ainsi été adaptés aux fins des expérimentations pertinentes à ce mémoire. Pour que ce traitement puisse être fait en direct comme dans

le cas d'un environnement changeant, avec présence humaine, de plus grandes ressources de traitement seraient nécessaires. L'algorithme de détection des collisions convergerait mieux et fournirait des résultats équivalents.

Enfin, puisque la méthode est très dépendante de la connaissance de la configuration du robot, conditionnant ainsi l'efficacité du simulateur qui utilise ces informations, une étude exhaustive de cette configuration serait d'une absolue nécessité.

Chapitre 3

A Constraint Based Kinematic Limitation Avoidance : The Sliding Algorithm For Six-Dimensional Collaborative Robots.

Résumé

Cet article présente un algorithme d'évitement de limitations cinématiques pour les robots collaboratifs. L'objectif principal de cette étude est d'améliorer la performance des utilisateurs et l'ergonomie d'utilisation du système lors d'interactions humain-robot. Présentement, dans de telles interactions, les utilisateurs humains travaillant avec des robots doivent tourner une bonne partie de leur attention vers le simple contrôle du robot plutôt que de se concentrer uniquement sur la tâche qu'ils veulent faire avec le robot. Effectivement, cette attention est requise pour éviter des problèmes tels que les limitations articulaires, les singularités et les collisions entre le robot et l'environnement immédiat. L'algorithme basé sur des contraintes géométrique proposé dans ce travail vise à régler ce problème. Cet algorithme, aussi appelé "sliding algorithm" (ou algorithme de glissement), permet à l'opérateur de détourner son attention des problèmes mentionnés plus haut en évitant automatiquement les limitations et en restant tout de même fidèle à la commande originalement envoyée par l'utilisateur. Premièrement, afin de bien décrire le fonctionnement de l'algorithme, ces fondations, en ce qui concerne le contrôle d'un robot UR5 à 6 degrés de liberté (DDL), sont présentées. Ensuite l'algorithme est présenté et détaillé spécifiquement pour l'implémentation sur un robot collaboratif à 6 DDL, soit le robot UR5. Finalement, des résultats d'expérimentations sont présentés afin de confirmer la performance de l'algorithme développé lorsque comparé à un algorithme d'évitement de collision bien établi dans le domaine, soit l'algorithme d'évitement de collision ressort-amortisseur virtuel.

Abstract

This paper presents a constraint-based kinematic limitation avoidance algorithm for collaborative robots. The main objective of this study is to improve the performance and intuitiveness of physical human-robot interactions. Currently, in such interactions, human users working with

robotic devices must divide their attention while performing tasks, focusing both on the task being performed and on the robot configuration. Indeed, the user must pay close attention to the robot in order to avoid issues such as joint position limitations, singularities and collisions with the environment. The proposed constraint-based algorithm, referred to as the "sliding algorithm", is designed to relieve the human user from having to monitor such limitations by automatically avoiding them while the system responds to the user's intentional commands. The framework, which was designed to manage several limitations occurring simultaneously in six-dimensional space (translation and orientation) is first presented. The algorithm is then presented and detailed for a specific six-degree-of-freedom collaborative serial robot, the UR5 from Universal Robots. Finally, experiments performed with human users to assess the algorithm's performance are described. These experiments evaluate different sliding options of the proposed algorithm against a classical virtual spring damper limitation avoidance algorithm by comparing the completion time of a given task and reviewing answers to subjective questions referring to the level of appreciation of the subjects.

3.1 Introduction

Collaborative and service robots are becoming more and more common in industrial and medical applications [6]. Serial robots, such as robotic arms, are one of a number of new collaborative robot technologies that are widely used to manipulate heavy objects, perform work in non-ergonomic environments, compensate for physical impairments or simply increase productivity. A major challenge in developing these devices is harmonizing the advantages they offer in terms of strength and endurance with the ability of human operators to adapt to unforeseen events. Overcoming this challenge would lead to significant improvements in human-robot interactions. Over the years, many robotic devices have been designed for a range of different tasks ; however, simplicity and intuitiveness of use have been recognized as important factors influencing the overall effectiveness of these systems [6], [7], [8]. Additionally, in order to be intuitive and predictable, the robot's motion should also be legible ; that is, the user should be able to quickly and confidently infer the robot's goal. This aspect was introduced and detailed in [9], [10], [8].

To allow for close cooperation with humans, safety measures must be in place to ensure users cannot make the robot collide with a nearby human or with itself. Currently, commercial robots have rudimentary strategies to mitigate these risks, such as stopping automatically when they detect any sign of an anomaly. There are also workarounds for singularities and joint limitations, which generally involve stopping the robot when an issue is encountered. However, these behaviors may appear non-intuitive to human users and can cause feelings of apprehension when operators encounter such situations. More importantly, if a singularity or joint limitation is encountered, the operator must pay close attention to the robot's architecture and try to move the robot away from it while simultaneously attempting to complete the intended task. The more degrees of freedom a robot has, the more complicated such operations may become.

Currently, a number of off-line path-planning strategies have been designed and self-collision avoi-

dance has been extensively studied [2], [3], [4], [5]. While the methods now in use are very efficient for certain tasks, they cannot be applied to the use case of a human operator controlling a robotic device. The reason these devices require a unique solution is that the trajectory must be adjusted at every time-step because no knowledge of the user's intention or the robot's final destination is available.

Many solutions have been proposed in the literature, such as potential fields [12], [13], [14] and virtual spring-damper systems [11], [15], and virtual-fixtures algorithms [16] have been implemented. The theoretical simplicity and versatility of these algorithms are their main advantage. In essence, the algorithms generate virtual forces that repel the robot from obstacles and other limitations. However, a number of drawbacks have been identified while implementing these methods. The algorithms require many finely adjusted parameters to work, and their efficiency depends on the robot architecture. For instance, a force applied at the base of a long link may not result in the same behavior as a force applied at its tip, which can result in counterintuitive behavior. In some cases a high degree of stiffness or repulsive potential field may induce vibrations or provoke instabilities [17], hindering the completion of the task and lowering the confidence level that an inexperienced user may have toward the robotic device. Finally, when navigating narrow channels, the robot may get stuck if the damping coefficient is set too high or the virtual mass of a point is set too low.

The proposed *constraint-based kinematic limitation avoidance algorithm*, also referred to as the *sliding algorithm*, addresses these issues by analytically computing the directions in which the robot may or may not move and by modifying the user's input according to the limitations encountered. The geometrical nature of the algorithm allows easier tuning of the few parameters required to implement it successfully. This approach was initially presented in [25] for a 2 DOF robot and extended in [22] to a 3 DOF robot. Each of these versions aimed at generalizing the algorithm in order to enable its implementation on any serial robot. The algorithm presented in this paper is an improvement and generalization of these methods to a six DOF system.

This paper is structured as follows. The core components of the *sliding algorithm*, such as the generation of the working environment, minimum distance and normal vector handling, and the application of geometric constraints are detailed. Then, the experimental results needed to verify the effectiveness of the *sliding algorithm* against the virtual spring damper algorithm are presented. Finally, the results are discussed and conclusions are drawn.

3.2 Algorithm

The algorithm is aimed at identifying limitations in real time (collisions, joint limitations, singularities) then modifying the Cartesian velocity input sent to the robot. The desired behavior can be described as *sliding* on these limitations. In this section, the algorithm is described and the various ways in which it can be implemented are detailed.

In order to extract the information from the environment and process it to correctly modify the velocity

input of the robot, six steps are required :

1. Representing the environment in which the robot operates in terms of proximity vectors normal to limitation planes.
2. Computing the robot's kinematics.
3. Generating the array of points on the robot architecture on which the closest distance will be computed.
4. Computing normal vectors of nearby obstacles in the end effector's referential.
5. Finding allowed sliding directions.
6. Applying geometric constraints to avoid the limitations.

These steps are explained below.

3.2.1 Generating the environment.

There are three types of limitations in the robot's workspace :

- Singularities
- Joint limitations
- Obstacles

In order to assess possible collisions with these limitations, it is necessary to represent them in a way that allows easy processing. The representation used for this work is a six-dimensional normal vector in the same Cartesian reference frame as the robot.

Singularities and Joint limitations

These two types of limitations can be treated in the same way. The joint limitation is expressed as a vector of angular speeds that must be respected in order to avoid the limitation. Then, the Jacobian at the configuration where the limitation occurs is computed. The normal vector $\vec{l}$ representing the limitation is given by the following equation :

$$\vec{l} = \frac{J\dot{\vec{\theta}}}{||J\dot{\vec{\theta}}||} \tag{3.1}$$

where J is the Jacobian matrix of the UR5 robot. As an example, the singularity of the UR5 robot at

$$\theta_5 = \{0, \pi\} \tag{3.2}$$

can be represented by the normal vector

$$\vec{\theta}_{Lim} = [0,0,0,0,\pm 1,0]^T \tag{3.3}$$

to the hyperplane generated by this singularity, in the joint space of the robot. This vector represents the direction in which the joints of the robot have to move in order to reach the singularity. The sign of this normal vector depends on the position of the robot's wrist. The normal Cartesian vector is then found by multiplying the Jacobian J with the angular velocity vector that moves the wrist towards $\vec{\theta}_{Lim}$:

$$\dot{\vec{\theta}} = [0,0,0,0,\mp 1,0]^T. \tag{3.4}$$

The norm of $\dot{\vec{\theta}}$ is arbitrarily set to 1 for simplicity, but one could use this property of the vector to convey information on how close the robot is to the singularity when considering the Jacobian conditioning. This limitation becomes active when joint 5 is closer to the singularity than a tolerance value set according to the frequency of the robot's controller and how quickly the condition number of the Jacobian matrix deteriorates near the singularity.

Objects

Object limitations are used to represent real 3-dimensional objects giving only three components (x-y-z) to the normal vectors representing the limitations. In some specific cases, objects with more than three Cartesian dimensions can be created in order to avoid certain velocities in a particular region of the workspace. An example of this use case could be a virtual object that limits rotations around the x and y axis to keep the end effector upright in a certain zone of the workspace.

3.2.2 Robot Kinematics

The Denavit-Hartenberg (DH) parameters of the UR5 robot are available in the literature, for example in [26]. While the DH parameters are useful to compute the robot kinematics, they are not well suited for our application. Because some of the joint reference frames are not located where the actual joint is, the architecture of the theoretical robot represented by the DH parameters is not well suited for direct kinematics computation of points located on links other than the end effector's link. By placing the frames differently, it is much simpler to easily compute direct kinematics for arbitrary points. As Figure 3.1 shows, it is not straightforward to adjust link lengths to quickly compute the forward kinematics of a point located on the third link, for instance. The resulting robot model is well suited to compute the end effector's kinematics along with the points on some of the wrist's links, but not for other links. Since the *sliding algorithm* is applied throughout the robot's links, not only at the end effector, it is important to be able to easily compute the forward kinematics and Jacobian matrix of a point located anywhere on the robot. By modifying the DH parameters it is possible to create a model that isolates each link, thus allowing computation of the kinematics and Jacobian matrix of a given point by simply modifying the length of the required links. The parameters used for the robot's kinematics are displayed in Figure 3.2 and the modified DH parameters are presented in Table 3.1.

3.2.3 Collision points

The collision points defined on the robot for the experiments are shown in Figure 3.3.

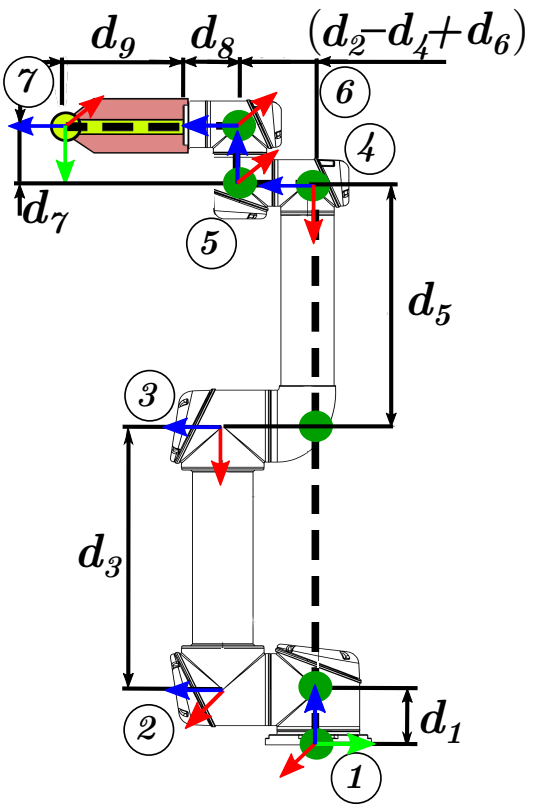

FIGURE 3.1 – Representation of the modified DH parameters used in the literature. Circled numbers identify the reference frame of each joint. Dark green dots are placed where the origins of the reference frames are located according to the DH parameters used. Red, green and blue arrows respectively identify the x, y, and z axis of each joint reference frame. The dashed line identifies the robot as it is described when the kinematics equations are solved with the chosen parameters.

TABLE 3.1 – Modified Denavit-Hartenberg parameters used to represent the UR5 robot

a_{i-1}	d_{i-1}	α_{i-1}	θ_i
0	d_1	$\frac{\pi}{2}$	θ_1
$-d_3$	d_2	0	θ_2
$-d_5$	$-d_4$	0	θ_3
0	d_6	$\frac{\pi}{2}$	θ_4
0	d_7	$\frac{-\pi}{2}$	θ_5
0	d_8+d_9	0	θ_6

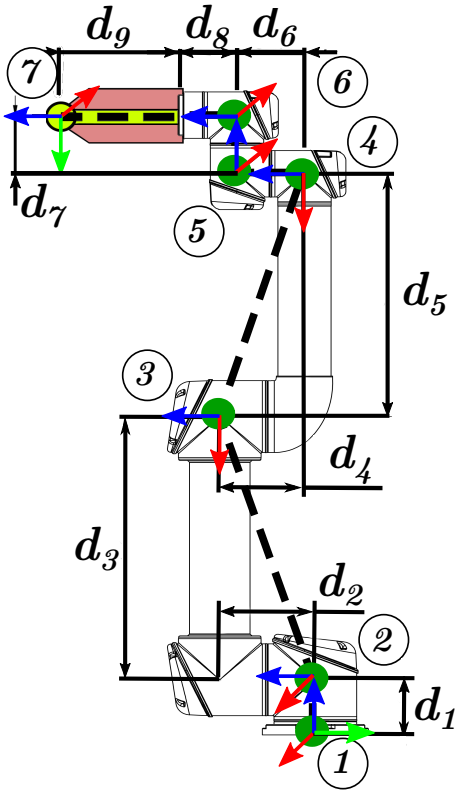

FIGURE 3.2 – Representation of the modified DH parameters used for this work. Circled numbers identify the reference frame of each joint. Dark green dots are placed where the robot joints are modeled according to the DH parameters used. Red, green and blue arrows respectively identify the x, y, z axis of each joint reference frame. The dashed line identifies the robot's mathematical model as it is described when the kinematics equations are solved with the chosen parameters.

These points are placed on the robot by changing the a_i and d_i values of the DH parameters according to the point's location on the links. This method can be directly applied on any robotic arm, provided the DH parameters closely represent the physical robot's architecture. Figure 3.3 exemplifies this process for points located between joint 3 and 4.

More points could be used to obtain better resolution, at the expense of larger computation time. This level of discretization proved to be well suited to the demands of the experiments. The collision detection algorithm is the same one used in [22], which allows minimum distance computation between points and 3D polygons.

The minimum distance from which a point can approach a limitation is the diameter of the robot link it is attached to. This approach is very similar to the approach proposed in [27].

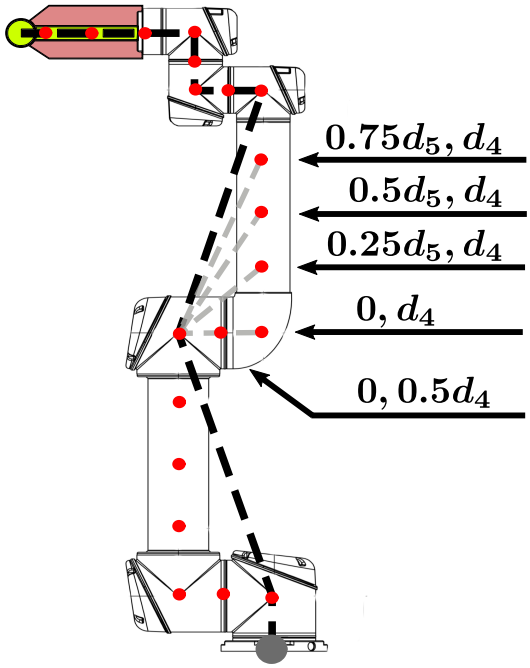

FIGURE 3.3 – Points at which the robot may collide. Arrows pointing toward a red dot are examples of how the DH parameters are sequentially modified to yield the direct kinematics and Jacobian matrices for every points.

3.2.4 Normal vectors from local reference frame to end-effector reference frame

In order to assess limitations and limit the motion commanded at the end effector, it is necessary to process these limitations simultaneously. The importance of this operation will be detailed in the section about geometric constraints. One way to achieve this is to transfer all limitations to the end effector before verifying the user command for possible collisions. Any limitation $\vec{l}_p$ encountered need to be transferred to a virtual limitation at the end effector $\vec{l}_{vp}$. The method used to find this virtual limitation is described in this section. It is possible to link the input end-effector velocity $\vec{v}$ by using the Jacobian matrix J, which maps the joint velocity vector,$\dot{\vec{\theta}}$, into the Cartesian and angular velocity, $\vec{v}$, as seen in equation 3.5.

$$\vec{v} = J\dot{\vec{\theta}}. \tag{3.5}$$

The same can be applied to the velocity of any point defined on the robot link. For example, consider the point P as seen in Figure 3.4.

The velocity of point P can be obtained as follows :

$$\vec{v}_p = J_p\dot{\vec{\theta}}. \tag{3.6}$$

where J_p is the Jacobian matrix defined at this point. Now the link between the end-effector velocity

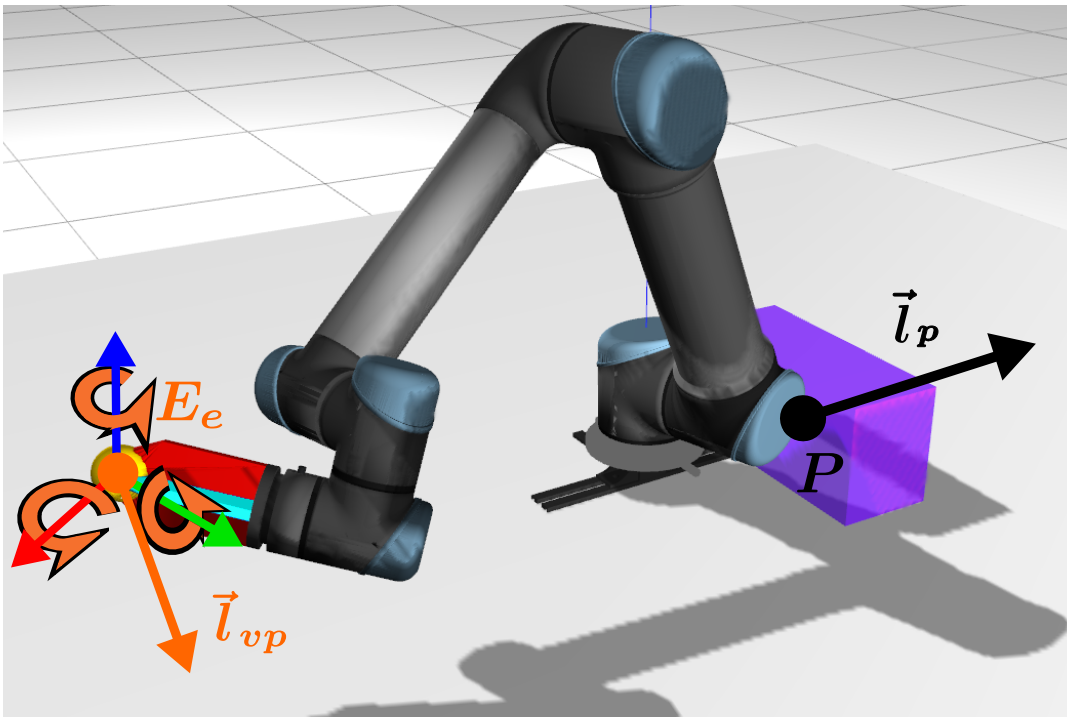

FIGURE 3.4 – The limitation $\vec{l}_p$ occurring at point P (in black), which has only up to three non-zero components in the x-y-z directions, is transfered to the end effector (point E_e) as a virtual limitation $\vec{l}_{vp}$ that now has up to 6 non-zero components (in orange).

and $\vec{v}_p$ can be easily obtained by combining equations 3.5 and 3.6 :

$$\vec{v}_p = J_p J^{-1} \vec{v}. \tag{3.7}$$

It is important to note the following details about last equation :

- Point P can be defined anywhere on the robot.
- The direct kinematics equations for P define system that has as many degrees of freedom as the number of joints between the robot's base and the point P.
- Consequently, the Jacobian matrix J_p may not be of full rank, removing the possibility of inverting it and directly resolving equation 3.7, the necessity for using another method arises.

Since velocity at point P must satisfy the limitation occurring at P, $\vec{l}_p$, the constraint

$$\vec{l}_p^T \vec{v}_p = 0 \tag{3.8}$$

is applied. The same can be done for the virtual limitation $\vec{l}_{vp}$ that will represent the limitation $\vec{l}_p$ at the end effector :

$$\vec{l}_{vp}^T \vec{v} = 0. \tag{3.9}$$

Using equations 3.7, 3.8, and 3.9, the following relation can be obtained if equations 3.7 and 3.9 are true for all values of $\vec{v}$:

$$\vec{l}_{vp}^T = \vec{l}_p^T J_p J^{-1}. \tag{3.10}$$

Rearranging, the result yields :

$$\vec{l}_{vp} = (J_p J^{-1})^T \vec{l}_p. \tag{3.11}$$

The described method allows to transfer all limitations $\vec{l}_p$, occurring at any given point on the robot, to virtual limitations $\vec{l}_{vp}$, now defined in the end effector's referential where the velocity is set at the desired rate by the user.

3.2.5 Finding allowed sliding directions

In [22], it was proposed to use cross products to find a perpendicular vector representing the direction in which the robot could move while satisfying two limitations. While this method works in three-dimensional space, it cannot be used in a six-dimensional space. Indeed, in $\mathscr{R}^6$ space, the cross product is not defined [28]. As demonstrated in [22], one cannot subtract simple projections of the input velocity of every limitation from the input velocity. The result of this operation will, in most cases, impose only one of the limitations if there are more than one encountered at a time. The solution proposed in this paper is to find a base of the nullspace of the matrix containing all encountered limitations. These vectors will be the allowed sliding directions since they will be orthogonal to all limitations. The method used to find the orthogonal complement of the set of limitations is a modified Gram-Schmidt process [17]. This process is aimed at building an orthogonal matrix from a given set of vectors regardless of their linear dependence between each other or if there are less vectors in the set than their dimension.

The process starts with a matrix comprising all limitations :

$$M_l = [\vec{l}_1 \ \vec{l}_2 \ ... \ \vec{l}_n] \tag{3.12}$$

with row vectors $\vec{l}_1 \ ... \ \vec{l}_n \in \mathscr{R}^6$. To spare computing time, a first verification can be done on the set of limitations by verifying the rank of the matrix. If

$$rank(M_l) = 6, \tag{3.13}$$

there are no components of the commanded velocity that can move the robot while respecting the limitations.

In the case where the rank is lower than 6, the process continues. By concatenating the identity matrix to M_l as follows, we ensure that the Gram-Schmidt process yields the missing orthogonal complement of the initial set of vectors :

$$V = [M_l \ I] \tag{3.14}$$

Then, the Gram-Schmidt process computes each column of the new orthogonalized matrix U by subtracting the projections of each vector of V upon all previously computed vectors of U. Each linearly dependant vectors of V will yield a zero vector on the corresponding column of U, ensuring that the maximum number of non-zero columns in U is equal to the number of dimensions of the space.

1. The projection function is defined as follows :

$$proj_{\vec{g}}(\vec{v}) = \frac{\vec{g}^T\vec{v}}{||\vec{g}||}\vec{g}. \tag{3.15}$$

   Where vector $\vec{v}$ is projected onto vector $\vec{g}$.

2. The Gram-Schmidt process yields a new set of vectors contained in matrix G. This set of vector, which contains the sliding vectors are computed :

$$\begin{aligned}
&\vec{g_1} = \vec{v}_1 \\
&\vec{g_2} = \vec{v}_2 - proj_{\vec{g}_1}(\vec{v}_2) \\
&\vec{g_3} = \vec{v}_3 - proj_{\vec{g}_1}(\vec{v}_3) - proj_{\vec{g}_2}(\vec{v}_3) \\
&\vdots \\
&\vec{g_k} = \vec{v}_k - \sum_{j=1}^{k-1} proj_{\vec{g}_j}(\vec{v}_k)
\end{aligned} \tag{3.16}$$

Columns of the matrix G that are linearly independent from all the limitations can be considered perpendicular to these limitations and will be the allowed sliding directions.

Example of the Gram-Schmidt process applied in a 6D space

Given the set of limitations M_L, find the set of sliding directions that respects the constraints.

$$M_L = \begin{bmatrix} 0.5774 & 0 & 0.4082 \\ 0 & 0.5774 & 0.4082 \\ 0.5774 & 0 & 0.4082 \\ 0 & 0.5774 & 0.4082 \\ 0.5774 & 0 & 0.4082 \\ 0 & 0.5774 & 0.4082 \end{bmatrix} \tag{3.17}$$

Since the rank of the matrix M_L is 2, it is possible to determine that the numbers of possible sliding vectors is 4 (6D space minus rank).

After removing the linearly-dependent limitations, the identity matrix is concatenated.

$$V = \begin{bmatrix} 0.5774 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.5774 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0.5774 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0.5774 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0.5774 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0.5774 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.18}$$

the Gram-Schmidt process is then applied to the matrix V. An example of the computation for the third column of the newly formed matrix follows :

$$\vec{u_3} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} - \begin{bmatrix} 0.3333 \\ 0 \\ 0.3333 \\ 0 \\ 0.3333 \\ 0 \end{bmatrix} - \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{3.19}$$

$$\vec{u_3} = \begin{bmatrix} 0.6666 \\ 0 \\ -0.3333 \\ 0 \\ -0.3333 \\ 0 \end{bmatrix} \tag{3.20}$$

The end result of the process gives the following vectors that are perpendicular to the set of limitations.

$$U = \begin{bmatrix} 0.8165 & 0 & 0 & 0 \\ 0 & 0.8165 & 0 & 0 \\ -0.4082 & 0 & 0.7071 & 0 \\ 0 & -0.4082 & 0 & 0.7071 \\ -0.4082 & 0 & -0.7071 & 0 \\ 0 & -0.4082 & 0 & -0.7071 \end{bmatrix} \tag{3.21}$$

It is now possible to use these vectors to slide on the limitations.

3.2.6 Applying geometric constraints.

The 6 DOF architecture allows different ways in which the input velocity can be modified to avoid limitations. Because translation and rotation constraints can influence one an other, some methods of avoiding limitations might be better suited for certain types of tasks. For instance, if the goal is to manipulate a glass of water while doing mostly translational trajectories, one may impose that the robot minimizes the magnitude of rotation velocities around the horizontal axes while sliding upon limitations to avoid spilling water.

Two of the many ways geometric constraints can be avoided have been investigated to assess their performance :

– The *Translation-Rotation commands sliding on Translation-Rotation constraints* ($TR-TR$) variant, where translation and rotation commands are both sliding on all constraints. In this variant, the robot will attempt to use all of its degrees of freedom to avoid any limitation.

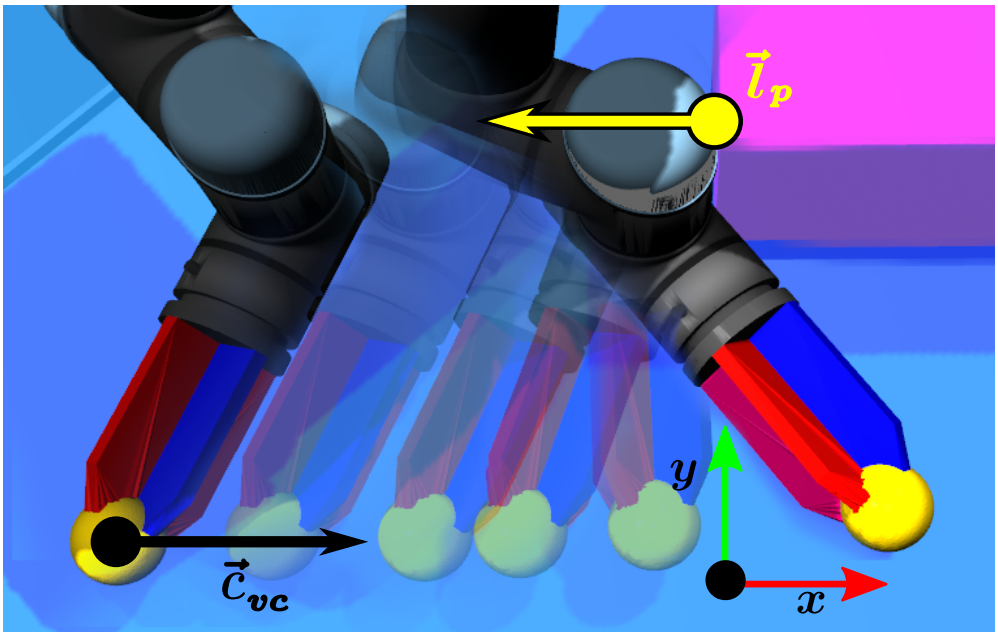

FIGURE 3.5 – $TR-TR$ variant : The robot encounters limitation $\vec{l}_p$ and then rotates, attempting to follow the user velocity command $\vec{u}_{vc} = [1,0,0,0,0,0]$.

– The *Translation command sliding on Translation constraints, Rotation commands sliding on Rotation constraints* ($T-T/R-R$) variant, where translation commands are able to slide only on translation constraints and rotation commands can only slide on rotation constraints. In this variant, the robot will only attempt to use its three translational degrees of freedom to avoid translation limitations and its three rotational degrees of freedom to avoid rotation limitations.

Many other options could be used but these two modes are more general and were selected for a first approach. Figures 3.5 and 3.6 help to illustrate these two concepts.

The desired velocity command sent by the user is then projected against each allowed sliding direction. For the $TR-TR$ variant, a velocity command $\vec{c} \in \mathscr{R}^6$ is directly projected onto the allowed sliding directions. The sum of these projections is the velocity that respects all of the limitations. The $T-T/R-R$ method splits the velocity command into a translation velocity command $\vec{c}_T$ and a rotation velocity command $\vec{c}_R$ which are projected onto translation and rotational limitations, respectively.

3.3 Experiments

This section presents the experimental protocol followed to asses the performance of the *sliding algorithm* and compare it to the virtual spring damper (V.S.D.) method [15]. The experiments were performed with a virtual UR5 robot modeled in Gazebo, a physics and visualization simulator embedded in ROS (Robot Operating System) [29]. The physical properties of the actuators and links were taken from the ROS universal_robot package [30]. In total, thirteen (13) human test subjects participa-

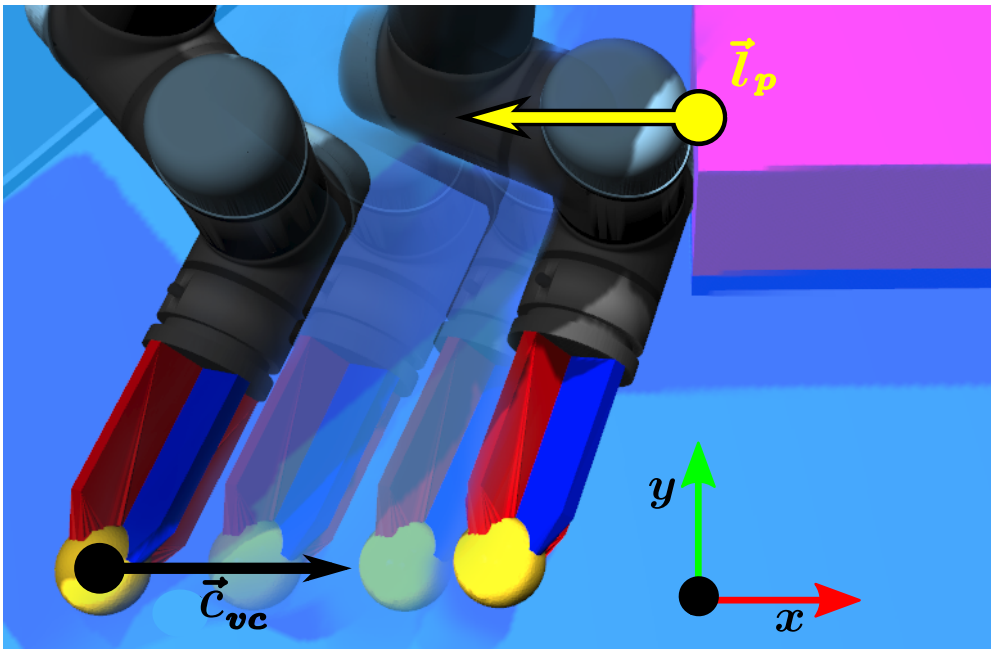

FIGURE 3.6 – $T-T/R-R$ variant : The robot encounters limitation $\vec{l}_p$ and slides on the $y-z$ plane while maintaining its initial orientation since the user velocity command $\vec{u}_{vc} = [1,0,0,0,0,0]$ is only a translation command.

ted in the experiments, which were approved by the ethics committee of Université Laval (certificate no. 2016-352 phase 2/25-09-2018).

3.3.1 Methodology

Since the objective of developing such algorithms is to improve the intuitive operation of devices and the performance of users, the participants were asked to test the algorithms and answer questions about their level of comfort and confidence when manipulating the robotic device. Two series of tests were run in the environment shown in Figure 3.7. The first series of tests contained three tests that examined both the $TR-TR$ and $T-T/R-R$ variants of the *sliding algorithm* and the V.S.D. algorithm, the later being used as a reference to evaluate the effectiveness of the *sliding algorithm*. These tests are referred to as *full-task tests*. The order in which the algorithms were tested was randomized, and participants were given time to practice with the controller before they began. These precautions were taken to avoid an algorithm being perceived as more efficient simply due to habituation. The task consisted of using the device to navigate towards each of the three zones displayed in Figure 3.7 with the end effector, in the specific order indicated. Participants answered a questionnaire based on the QUEAD [31] after using each of the three algorithms. These questions concerned the user's comfort level while using each algorithm, the perceived efficiency while completing the task, the ease with which the participants could predict the motion of the robot and their overall appreciation of the help the algorithm provided in order to complete the task.

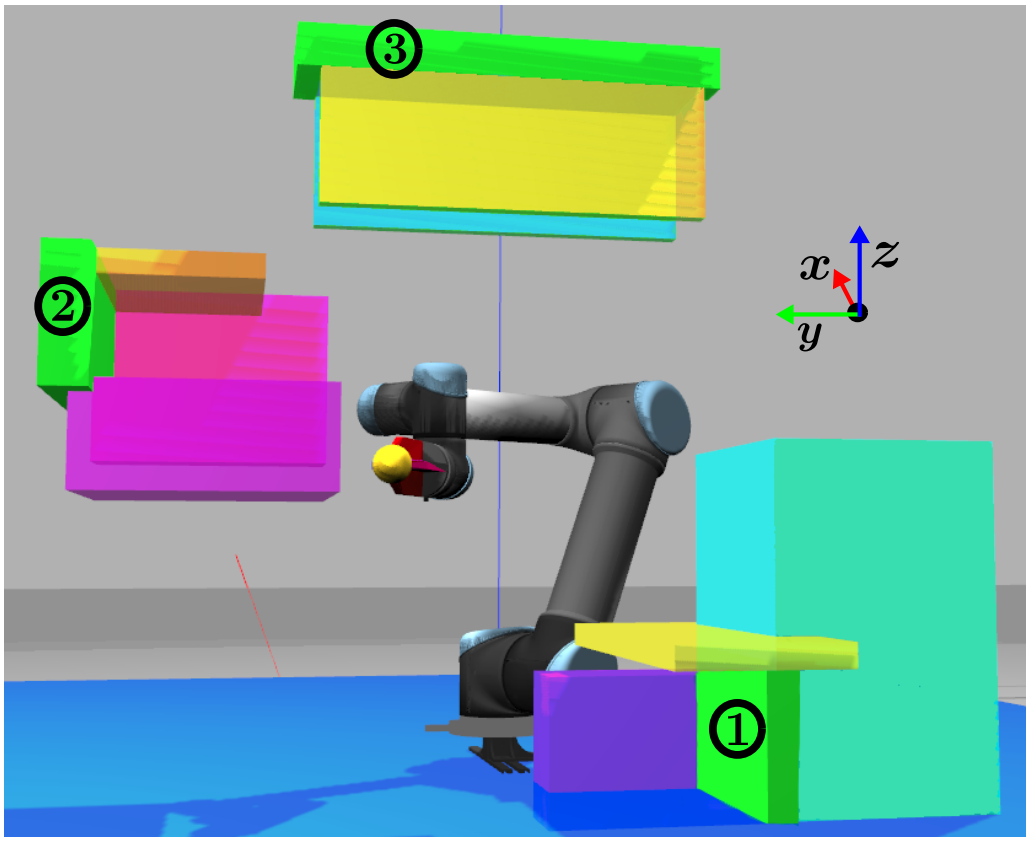

FIGURE 3.7 – Environment used for the experiments.

The last series of tests contained two tests that examined participants' ability to perform a shortened version of the task with the $TR-TR$ variant and use of the *stop when colliding* algorithm commonly used in the industry. These tests are referred to as *partial-task tests*. The shortened task consisted of simply reaching the first zone displayed in Figure 3.7. A simpler task was chosen because the 6 DOF *UR5* robot is quite complicated to control, and this would have made the full-length experiments too arduous, even for experienced users. The *stop when colliding* algorithm represents a behavior that is used for protection zones in collaborative environments [32] and employed as a base-line to asses the performance of the *sliding algorithm*.

The time required to complete each test is used as an objective measure to assess the performance of the algorithms.

3.3.2 The task

The task given to the test subjects was specifically defined to assess the performance of the algorithms in avoiding objects in the environment, as well as navigating near singularities. When a participant is required to navigate from the start position (the robot's position in Figure 3.7) to the three targets, the user has to move the robot near its wrist singularity, defined by equation 3.2, where the end effector's

TABLE 3.2 – Average time and standard deviation, in seconds, for each algorithm used in the full-task tests.

Algorithm	Average completion time	Standard deviation
$T-T/R-R$	127.38s	46.4s
$TR-TR$	69.2s	26.6s
V.S.D	93.37s	32.7s

link is aligned with the first link of the wrist. The performance of each algorithm in effectively reproducing the user's intention while avoiding a singularity is thus evaluated. It is particularly important to gauge this capacity, since singularities can be hard to mentally visualize, limiting the user's ability to consciously avoid them.

3.3.3 Results : Full-task tests

The average completion time and standard deviation of the three tests are presented in Table 3.2. The time required to complete any task with a robotic device is highly dependent on a user's ability to control 3D rotations along the robot's base axis referential and to control the 6 DOF with a standard joystick. Because of this, it is important to compare participants' task completion times using the $TR-TR$ and $T-T/R-R$ variants to their results using the V.S.D., the control algorithm for this experiment. A Friedman test [33] run on the three experiments yielded $p = 1.737 * 10^{-5} < 0.05$, which allows to conclude that the full-task tests can be used to assess the performances of the algorithms. Figure 3.8 shows the relative performance of the $TR-TR$ and $T-T/R-R$ variants against the V.S.D. algorithm.

TR-TR variant performance analysis

The time difference between the $TR-TR$ variant and the V.S.D algorithm is considered significant according to a paired two-sample sign test ($p = 0.00073 < 0.05$). Considering the best result, performance using the $TR-TR$ algorithm, one can observe a 25% relative improvement in task performance time when compared to the V.S.D. algorithm. The standard deviation of the $TR-TR$ algorithm test is also lower than the V.S.D. algorithm test, suggesting that more participants were able to consistently benefit from the help the algorithm provided. During the tests, it was observed, from participant's comments, that the V.S.D. algorithm's main drawback was the repulsion forces, which were not affecting the robot in the same way as the object limitations. In some cases, the robot was translated away from the singularity in a disproportionate manner (30 to 40cm away from the original location) until it reached an other obstacle when the user input a command to manually try to move the robot away from the singularity. By observation, this may be due to the magnitude of each of the components of the normal vector. If the rotation components are too small, the V.S.D. algorithm may create a repulsive force from a singularity that mostly consists of translational components. Translation commands are not an efficient means of achieving this goal since the translations and rotation commands are

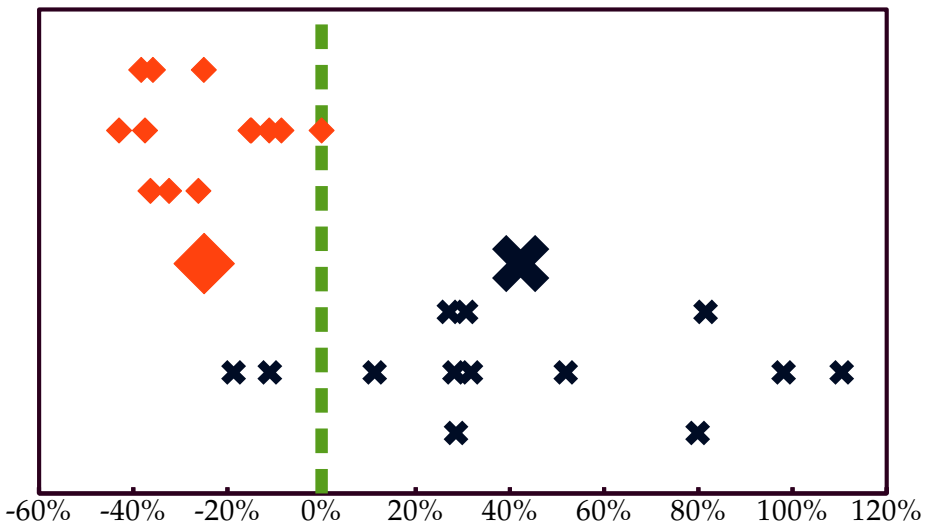

FIGURE 3.8 – Relative time difference for the $TR-TR$ variant, represented with orange diamonds, and the $T-T/R-R$ variant, represented with black crosses, using the V.S.D. algorithm for reference, represented by the green dashed line. Large markers indicate average times.

almost decoupled from each other due to the robot's architecture. The resulting behavior is that the robot is pushed away in a disproportionate manner relative to the behavior the user has experienced while avoiding objects. In contrast, the behavior of the $TR-TR$ algorithm in this regard is to mainly rotate around the singularity until the wrist's rotation axis has flipped directions, as shown in Figure 3.9. The last frame of this figure shows the robot arm moving away from the singularity because the rotation axis of joint 5 has completely changed direction with respect to the robot's global referential.

T-T/R-R variant performance analysis

The difference in task completion time between the $T-T/R-R$ variant and V.S.D algorithm is considered significant according to a paired two-sample sign test ($p = 0.022 < 0.05$). On average, the participants performed the series 42% faster with the V.S.D. algorithm than with the $T-T/R-R$ variant. In some instances, participants with previous experience navigating in a 6 DOF environment, such as previous experience using 3D rendering software, 3D CAD software, or video games, found the $T-T/R-R$ easier to manipulate. This may be due to the fact the this algorithm does not orient the end effector in the direction of translation when colliding with an object. This allowed more freedom while controlling multiple degrees of freedom simultaneously. Otherwise, the majority of participants who benefited more from the $TR-TR$ variant or the V.S.D algorithm tended to only control 1 degree of freedom at a time. The most time-consuming aspects of the tests generally occurred when complex rotations where needed to align the device in a certain way to attain an objective.

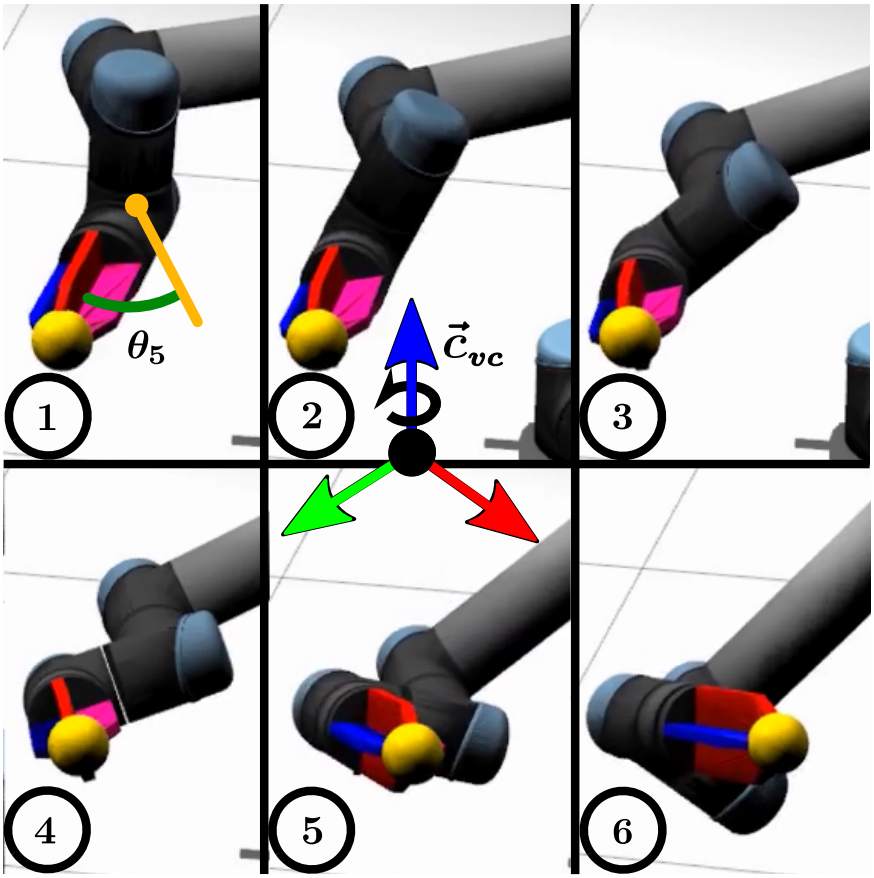

FIGURE 3.9 – Behavior of the $TR-TR$ variant while moving directly toward the wrist singularity at θ_5. $\vec{c}_{vc}$ is the user's velocity command during the whole process.

TABLE 3.3 – Average score for QUEAD questionnaire of the full-task tests based on the seven-point Likert scale (scores from 1 to 7, higher is better).

Algorithm	Average score
$T-T/R-R$	4.09/7
$TR-TR$	5.85/7
V.S.D	4.67/7

3.3.4 Subjective results : full-task tests

After each experiment, participants rated their experiences with the algorithms by answering 5 questions (1, 3, 9, 11, 13) taken from the QUEAD [31]. The scores attained by each algorithm, based on the seven-point Likert scale, offer an insight into the participants' positive or negative opinions about the control modes, as shown in Table 3.3.

The scores demonstrate that study participants had positive experiences using both the $TR-TR$ and V.S.D. algorithms. The participants were generally neutral toward the $T-T/R-R$ algorithm.

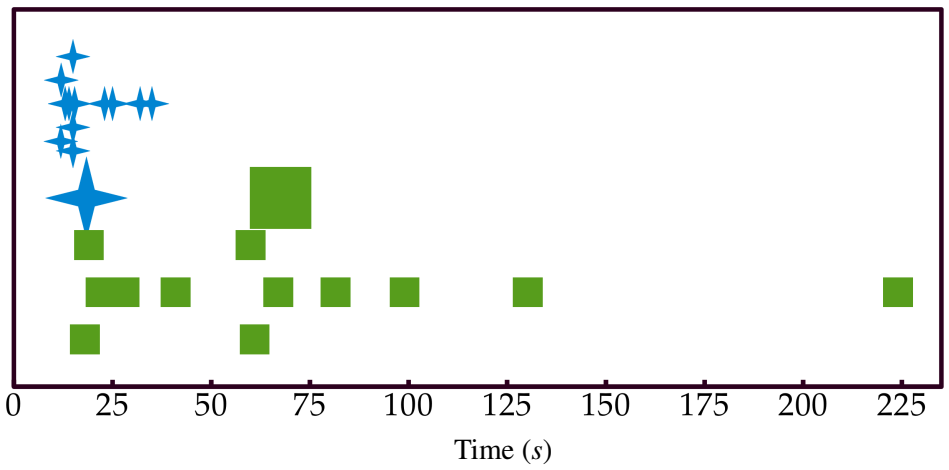

FIGURE 3.10 – Results, in seconds, of the partial tests done with the $TR-TR$ variant and the *stop when colliding* algorithm. Big markers indicate averages.

3.3.5 Objective results : partial task tests

The average time taken to accomplish the partial-task test with the $TR-TR$ variant is 18.4 seconds with a standard deviation of 7.8 seconds. The same task done with the *stop when colliding* algorithm took an average of 67.6 seconds with a standard deviation of 58 seconds. The time difference of 49.2 seconds (72.8% time decrease) is considered significant according to the Mann-Whitney-Wilcoxon non-parametric one-tailed test ($p = 0.002 < 0.05$). Figure 3.10 shows data from the *stop when colliding* algorithm are displayed as green squares, the $TR-TR$ variant results are displayed as blue stars. Big markers indicate averages. It can also be noted that some of the results from the *stop when colliding* algorithm dataset are in the same range as the $TR-TR$ variant range. This allows us to verify that it was possible, but much harder, to accomplish the tasks at a reasonable rate with the *stop when colliding* algorithm. Finally, the last two observations allow us to infer that use of the $TR-TR$ algorithm not only increases the performance times for the majority of users, but also makes operation of the robot intuitive enough for users who lack robotics experience and can bring their performance in line with more experienced users.

3.4 Conclusion

The main goal of the research presented in this paper was to improve on previous work focusing on similar algorithms and to compare the results of the proposed sliding algorithm to state-of-the-art approaches such as the virtual spring damper algorithm. Experiments were performed to observe whether relative efficiency gains were achieved when users completed tasks with the help of the sliding algorithm as compared to the V.S.D. algorithm. Two different methods of implementing the *sliding algorithm* were identified as having the potential to be efficient for the task tests : the $TR-TR$ variant and the $T-T/R-R$ variant. Experiments revealed an average completion time decrease of about 25% with the $TR-TR$ and a 42% time increase for the $T-T/R-R$ variant when compared to the time

required to complete the same tasks with the V.S.D. algorithm. Another experiment was conducted with the $TR-TR$ algorithm to compare the performance of this variant against the behavior widely encountered on robotic arms, i.e., the *stop when colliding* algorithm. The average time improvement for task completion between these two algorithms was 72.8% in favor of the $TR-TR$ variant. These results indicate that the proposed *sliding algorithm* improves user performance when the $TR-TR$ variant is applied, but no performance gain was noted using the $T-T/R-R$ variant. In addition, users with no prior training or experience operating robotic devices were able to complete the tasks efficiently with the proposed algorithm while it was difficult with VSD or stop when colliding. Further work examining interactions with moving objects would improve this algorithm. Also, in the case of dealing with unforeseen objects in the environment, extending the algorithm's use to include handling data from haptic or vision sensors could be a good way to increase its versatility. Finally, future work may include exploring augmented reality as a means of enabling visualization of limitations such as joint limitations and singularities. This could enhance the performance of users, who normally have no knowledge of the existence of such limitations prior to encountering them.

Conclusion

Les articles présentés dans ce mémoire ont fait état du développement d'un algorithme de cinématique d'interaction dans un contexte de coopération humain-robot.

En effet l'algorithme préliminaire implémenté sur un robot sériel à 3 degrés de liberté a été présenté. Des expérimentations ont démontré que l'algorithme pouvait bel et bien améliorer les performances des utilisateurs comparativement à l'algorithme présentement répandu en industrie. Une brève comparaison à aussi été faite avec un autre type d'algorithme, soit l'algorithme ressort-amortiseur virtuel. Différentes problématiques ont été identifiées en rapport avec l'utilisation de cet algorithme lorsque comparé aux performances de l'algorithme de cinématique d'interaction développé.

Ensuite des améliorations entourant l'implémentation de l'algorithme de cinématique d'interaction sont présentées. Un algorithme de détection de collisions et de distances minimales entre des objets en 3 dimensions de forme convexe est implémenté et permet la gestion de formes plus complexes lors de l'évitement de collisions. Une technique de balayage de l'environement de travail avec l'aide d'une caméra Kinect, suivi du post traitement d'un nuage de points et d'une procédure de segmentation en polygones convexes est utilisée pour automatiser la génération de la représentation virtuelle de l'environnement de travail. Mises en commun, ces techniques représentent un bon exemple de mise en oeuvre de l'algorithme de cinématique d'interaction malgrés l'identification de certaines imperfections au niveau de l'algorithme de détection de collisions.

Finalement, l'algorithme de cinématique d'interaction est généralisé au cas d'un robot sériel à 6 degrés de liberté. Les modifications majeures apportées à l'algorithme par rapport à la version antérieure de l'algorithme à 3 degrés de liberté sont présentées. L'utilisation du procédé de Gram-Schmidt requis pour respecter les contraintes de déplacement est détaillée ainsi que les modifications des paramètres de Denavit–Hartenberg pour la génération du modèle mathématique gérant le déplacement du robot. Pour conclure, des expérimentations ont été menées pour valider le gain de performance lorsque l'algorithme est utilisé dans différents contextes. Une comparaison objective avec l'algorithme de ressort-amortisseur virtuel a aussi été réalisée. Au cours de cette comparaison, l'algorithme de cinématique d'interaction, lorsqu'utilisé sous la variante $TR-TR$, s'est avéré 25% plus performant en terme de temps requis pour terminer une tâche donnée. Un questionnaire suivant les expérimentations a aussi révélé que l'algorithme de cinématique d'interaction est plus facile et intuitif à utiliser que l'autre algorithme testé lors de l'expérimentation.

Futures avancées

Différentes facettes de ce projet de recherche peuvent être approfondies. Voici quelques idées qui auraient le potentiel d'améliorer l'algorithme ou la facilité avec laquelle l'utilisateur peut interagir avec un bras robotique sur lequel l'algorithme d'interaction serait implémenté.

1. L'algorithme de cinématique d'interaction, sous sa forme actuelle, requiert une calibration manuelle d'une distance de sécurité, en terme de position articulaire, lorsque près d'un singularité avant que la limitation prévenant l'approche du robot soit activée. Ceci est peu ergonomique et requiert l'attention d'une personne ayant de bonnes connaissances en robotique. Le développement d'une méthode pour évaluer automatiquement la dégradation du conditionnement de la matrice Jacobienne permettrait la simplification de la mise en place de l'algorithme.
2. La principale cause de perte de temps durant de la réalisation de la tâche demandée lors des expérimentations est rencontrée lors des interactions avec des limitations non visibles par les utilisateurs telles que les singularités et les limitations articulaires. Ces limitations, bien que plus faciles à gérer avec l'algorithme que sans, posent encore problème puisqu'elles font en sorte que le robot dérive de la trajectoire commandée par l'utilisateur sans qu'il ne sache qu'il est présentement près d'une limitation. L'utilisation de la technologie de la réalité augmentée pourrait permettre la visualisation en temps réel de ces limitations, ce qui permettrait à l'utilisateur d'anticiper la réaction du robot.

Lien avec les travaux du laboratoire

Les travaux présentés dans ce mémoire sont en lien avec différents travaux effectués au cours des dernières années en interaction humain-robot et en ingénierie de la réadaptation au laboratoire de robotique de l'Université Laval et au Centre interdisciplinaire de recherche en réadaptation et intégration sociale (CIRRIS). Afin de mettre les travaux de ce mémoire dans ce contexte et de donner des références au lecteur par rapport à ces travaux afin de continuer ses lectures, voici une courte mise en contexte.

Au cours des dernières années, différents travaux ont été entrepris au laboratoire de robotique dans le domaine de l'interaction physique humain robot. Les travaux présentés dans ce mémoire s'inscrivent dans ce contexte et pourraient aider des opérateurs en industrie à opérer de manière plus efficace les robots collaboratifs. Les travaux en lien avec ce contexte au laboratoire de robotique de l'Université Laval sont présentés. Des travaux ont été effectués au niveau de systèmes d'assistances intelligents complets [34, 35, 36, 37, 38, 39, 40, 41], des modélisations théoriques [42], ainsi que des algorithmes intelligents permettant d'améliorer les performances des systèmes d'interaction, [43, 44, 45, 46, 47, 48, 49].

Les travaux s'inscrivent également dans le domaine de l'ingénierie de la réadaptation qui a récemment pris un essor au laboratoire de robotique et au CIRRIS. L'objectif global de ces travaux est de permettre

aux personnes vivant avec des incapacités physiques d'accomplir différentes activités de la vie quotidienne de manière autonome et ainsi accroître leur qualité de vie. Les travaux portent sur le développement de mécanismes d'assistance [50, 51, 52], d'algorithmes intelligents [53, 54, 55, 56, 57, 22, 58] et d'interfaces de contrôle [59, 60, 61, 62, 63, 64, 65, 66, 67, 68] ainsi que l'évaluation de ces technologies [69, 70, 71].

Bibliographie

[1] C. Ericson, *Real-Time Collision Detection*. 2005.

[2] S. R. S. Buss, "Introduction to inverse kinematics with jacobian transpose, pseudoinverse and damped least squares methods," *IEEE Journal of Robotics and Automation*, vol. 17, no. 4, pp. 1–19, 2004.

[3] S. R. Buss and J.-S. Kim, "Selectively Damped Least Squares for Inverse Kinematics," *Journal of Graphics, GPU, and Game Tools*, vol. 10, no. 3, pp. 37–49, 2005.

[4] D. E. Schinstock, "Approximate solutions to unreachable commands in teleoperation of a robot," *Robotics and Computer-Integrated Manufacturing*, vol. 14, no. 3, pp. 219–227, 1998.

[5] J. Kuffner and K. Nishiwaki, "Self-Collision Detection and Prevention for Humanoid Robots.," *Proceedings of the IEEE International Conference on Robotics & Automation,*, pp. 2265–2270, 2002.

[6] M. Vagaš, J. Semjon, M. Hajduk, L. Páchniková, and M. Lipčak, "The view to the current state of robotics," *Proceedings of 2011 International Conference on Optimization of the Robots and Manipulators*, 2011.

[7] A. Lecours, B. Mayer-St-Onge, and C. Gosselin, "Variable admittance control of a four-degree-of-freedom intelligent assist device," *Proceedings - IEEE International Conference on Robotics and Automation*, no. 2, pp. 3903–3908, 2012.

[8] A. D. Dragan, S. Bauman, J. Forlizzi, and S. S. Srinivasa, "Effects of Robot Motion on Human-Robot Collaboration," *ACM/IEEE Int. Conference on Human-Robot Interaction*, pp. 51–58, 2015.

[9] A. Campeau-Lecours and C. Gosselin, "An anticipative kinematic limitation avoidance algorithm for collaborative robots : Two-dimensional case," *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 4232–4237, 2016.

[10] A. De Santis, B. Siciliano, A. De Luca, and A. Bicchi, "An atlas of physical human-robot interaction," *Mechanism and Machine Theory*, vol. 43, no. 3, pp. 253–270, 2008.

[11] F. Seto, K. Kosuge, and Y. Hirata, "Self-collision Avoidance Motion Control for Human Robot Cooperation System using RoBE," *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 3143–3148, 2005.

[12] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," *Proceedings of the IEEE International Conference on Robotics and Automation*, vol. 2, pp. 500–505, 1985.

[13] S. Haddadin, R. Belder, and A. Albu-Schäffer, "Dynamic Motion Planning for Robots in Partially Unknown Environments," *IFAC World Congress (IFAC2011), Milano, Italy*, vol. 18, 2011.

[14] Y. Koren and J. Borenstein, "Potential Field Methods and Their Inherent Limitations for Mobile Robot Navigation," *IEEE Int. Conference on Robotics and Automation*, pp. 1398–1404, 1991.

[15] A. De Santis, A. Albu-Schäffer, C. Ott, B. Siciliano, and G. Hirzinger, "The skeleton algorithm for self-collision avoidance of a humanoid manipulator," *IEEE/ASME International Conference on Advanced Intelligent Mechatronics, AIM*, pp. 1–6, 2007.

[16] L. B. Rosenberg, "Perceptual tools for Telerobotic Manipulation," *Proceedings of IEEE Virtual Reality Annual International Symposium*, pp. 76–82, 1993.

[17] F. Flacco, T. Kröger, A. De Luca, and O. Khatib, "A Depth Space Approach to Human-Robot Collision Avoidance," *IEEE International Conference on Robotics and Automation*, pp. 338–345, 2012.

[18] G. V. D. Bergen, "A Fast and Robust GJK Implementation for Collision Detection of Convex Objects," *Journal of Graphics Tools*, vol. 4, no. 2, pp. 7–25, 1999.

[19] H. Samet, "Spatial Data Structures : Quadtree, Octrees and Other Hierarchical Methods.," *Symposium on Large Spatial Databases*, pp. 191–212, 1989.

[20] G. Liu, Z. Xi, and J. M. Lien, "Nearly convex segmentation of polyhedra through convex ridge separation," *CAD Computer Aided Design*, vol. 78, pp. 137–146, 2016.

[21] V. Maheu, J. Frappier, P. S. Archambault, and F. Routhier, "Evaluation of the JACO robotic arm : Clinico-economic study for powered wheelchair users with upper-extremity disabilities," in *2011 IEEE International Conference on Rehabilitation Robotics*, vol. 34, pp. 1–5, IEEE, jun 2011.

[22] P. LeBel, C. Gosselin, and A. Campeau-Lecours, "An anticipative kinematic limitation avoidance algorithm for collaborative robots : Three-dimensional case," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 3075–3080, IEEE, sep 2017.

[23] W. E. Red, "Minimum distances for robot task simulation," *Robotica*, vol. 1, pp. 231–238, 1983.

[24] E. G. Gilbert, D. W. Johnson, and S. S. Keerthi, "A Fast Procedure for Computing the Distance Between Complex Objects in Three-Dimensional Space," *IEEE Journal on Robotics and Automation*, vol. 4, no. 2, pp. 193–203, 1988.

[25] A. Campeau-Lecours, V. Maheu, S. Lepage, H. Lamontagne, S. Latour, L. Paquet, and N. Hardie, "JACO Assistive Robotic Device : Empowering People With Disabilities Through Innovative Algorithms," *Rehabilitation Engineering and Assistive Technology Society of North America (RESNA) Annual Conference*, 2016.

[26] M. R. de Gier, "Control of a robotic arm - Application to on-surface 3D-printing," *Master@Delft University of Technology*, vol. 13, pp. 1–13, 2009.

[27] M. Selvaggio, F. Abi-farraj, C. Pacchierotti, P. R. Giordano, and B. Siciliano, "Haptic-Based Shared-Control Methods for a Dual-Arm System," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 4249–4256, 2018.

[28] W. S. Massey, "Cross Product of Vektors in higher Dimensional Euclidean spaces," *The American Mathematical Monthly*, vol. 90, no. 10, pp. 697–701, 1983.

[29] ROS, "ROS.org | Powering the world's robots," 2018.

[30] S. Edwards, S. Glaser, K. Hawkins, W. Meeussen, and F. Messmer, "universal_robot - ROS Wiki," 2018.

[31] J. Schmidtler, K. Bengler, F. Dimeas, and A. Campeau-Lecours, "A Questionnaire for the Evaluation of Physical Assistive Devices (QUEAD)," *IEEE International Conference on Systems, Man, and Cybernetics*, no. March, 2017.

[32] G. Michalos, S. Makris, P. Tsarouchi, T. Guasch, D. Kontovrakis, and G. Chryssolouris, "Design considerations for safe human-robot collaborative workplaces," *Procedia CIRP*, vol. 37, pp. 248–253, 2015.

[33] W. Daniel, *Applied nonparametric statistics*. The Duxbury advanced series in statistics and decision sciences, PWS-Kent Publ., 1990.

[34] C. Gosselin, T. Laliberté, B. Mayer-St-Onge, S. Foucault, A. Lecours, V. Duchaine, N. Paradis, D. Gao, and R. Menassa, "A friendly beast of burden : A human-assistive robot for handling large payloads," *IEEE Robotics and Automation Magazine*, vol. 20, no. 4, pp. 139–147, 2013.

[35] A. Campeau-Lecours, "Développement d'algorithmes de commande et d'interfaces mécatroniques pour l'interaction physique humain-robot," 2012.

[36] A. Campeau-Lecours, S. Foucault, T. Laliberté, B. Mayer-St-Onge, and C. Gosselin, "A cable-suspended intelligent crane assist device for the intuitive manipulation of large payloads," 2016.

[37] A. Campeau-Lecours, P.-L. Belzile, T. Laliberté, S. Foucault, B. Mayer-St-Onge, D. Gao, and C. Gosselin, "An articulated assistive robot for intuitive hands-on-payload manipulation," *Robotics and Computer-Integrated Manufacturing*, vol. 48, pp. 182–187, 2017.

[38] D. Gao, A. Lecours, T. Laliberte, S. Foucault, C. Gosselin, B. Mayer-St-Onge, R. J. Menassa, and P.-L. Belzile, "Movement system configured for moving a payload," 2017.

[39] A. Lecours, B. Mayer-St-Onge, C. Gosselin, and D. Gao, "Method of inferring intentions of an operator to move a robotic system," 2016.

[40] A. Lecours, S. Foucault, T. Laliberte, C. Gosselin, B. Mayer-St-Onge, D. Gao, and R. J. Menassa, "Movement system configured for moving a payload in a plurality of directions," 2015.

[41] D. Gao, D. M. Wegner, R. J. Menassa, A. Lecours, C. Gosselin, T. Laliberte, S. Foucault, and V. Duchaine, "Sensor for handling system," 2014.

[42] A. Campeau-Lecours, M. J. D. Otis, and C. Gosselin, "Modeling of physical human–robot interaction : Admittance controllers applied to intelligent assist devices with large payload," *International Journal of Advanced Robotic Systems*, vol. 13, no. 5, p. 1729881416658167, 2016.

[43] M. A. Sassi, M. J. D. Otis, and A. Campeau-Lecours, "Active stability observer using artificial neural network for intuitive physical human–robot interaction," *International Journal of Advanced Robotic Systems*, vol. 14, no. 4, p. 1729881417727326, 2017.

[44] A. Campeau-Lecours, M. Otis, P.-L. Belzile, and C. Gosselin, "A time-domain vibration observer and controller for physical human-robot interaction," *Mechatronics*, vol. 36, pp. 45–53, 2016.

[45] A. Lecours and C. Gosselin, "Computed-torque control of a four-degree-of-freedom admittance controlled intelligent assist device," in *Experimental Robotics*, pp. 635–649, Springer, 2013.

[46] A. Lecours, B. M. St-Onge, and C. Gosselin, "Variable admittance control of a four-degree-of-freedom intelligent assist device.," in *ICRA*, pp. 3903–3908, 2012.

[47] P. D. Labrecque, J.-M. Haché, M. Abdallah, and C. Gosselin, "Low-impedance physical human-robot interaction using an active–passive dynamics decoupling," *IEEE Robotics and Automation Letters*, vol. 1, no. 2, pp. 938–945, 2016.

[48] P. D. Labrecque, T. Laliberté, S. Foucault, M. E. Abdallah, and C. Gosselin, "uMan : A low-impedance manipulator for human–robot cooperation based on underactuated redundancy," *IEEE/ASME Transactions on Mechatronics*, vol. 22, no. 3, pp. 1401–1411, 2017.

[49] P. D. Labrecque and C. Gosselin, "Variable admittance for pHRI : from intuitive unilateral interaction to optimal bilateral force amplification," *Robotics and Computer-Integrated Manufacturing*, vol. 52, pp. 1–8, 2018.

[50] G. Lemire, T. Laliberte, V. Flamand, P. Cardou, and A. Campeau-Lecours, "Preliminary design of a writing assistive device for people living with uncoordinated movements," *Rehabilitation Engineering and Assistive Technology Society of North America (RESNA)*, 2019.

[51] P. Turgeon, T. Laliberte, F. Routhier, V. Flamand, and A. Campeau-Lecours, "Preliminary design of an active stabilization assistive eating device for people living with uncoordinated movements," in *Rehabilitation Robotics (ICORR), 2019 International Conference on*, IEEE, 2019.

[52] P. Turgeon, M. Dube, T. Laliberte, P. Archambault, V. Flamand, F. Routhier, and A. Campeau-Lecours, "Mechanical design of a new assistive eating device for people living with movement disorders," *Assistive Technology Journal*, 2019.

[53] D.-S. Vu, U. C. Allard, C. Gosselin, F. Routhier, B. Gosselin, and A. Campeau-Lecours, "Intuitive adaptive orientation control of assistive robots for people living with upper limb disabilities," in *Rehabilitation Robotics (ICORR), 2017 International Conference on*, pp. 795–800, IEEE, 2017.

[54] A. Campeau-Lecours, U. Côté-Allard, D.-S. Vu, F. Routhier, B. Gosselin, and C. Gosselin, "Intuitive adaptive orientation control for enhanced human-robot interaction," *IEEE Transactions on robotics*, 2019.

[55] A. Campeau-Lecours, H. Lamontagne, S. Latour, P. Fauteux, V. Maheu, F. Boucher, C. Deguire, and L.-J. C. L'Ecuyer, "Kinova Modular Robot Arms for Service Robotics Applications," *International Journal of Robotics Applications and Technologies (IJRAT)*, vol. 5, no. 2, pp. 49–71, 2017.

[56] A. Campeau-Lecours, V. Maheu, S. Lepage, H. Lamontagne, S. Latour, L. Paquet, and N. Hardie, "Jaco assistive robotic device : Empowering people with disabilities through innovative algorithms," *Rehabilitation Engineering and Assistive Technology Society of North America (RESNA)*, 2016.

[57] A. Campeau-Lecours and C. Gosselin, "An anticipative kinematic limitation avoidance algorithm for collaborative robots : Two-dimensional case.," in *IROS*, pp. 4232–4237, 2016.

[58] P. LeBel, C. Gosselin, and A. Campeau-Lecours, "A Constraint Based Kinematic Limitation Avoidance : The Sliding Algorithm For Six-Dimensional Collaborative Robots," *IEEE Transactions on robotics*, 2019.

[59] S. Poirier, U. Cote-Allard, F. Routhier, and A. Campeau-Lecours, "Voice Control Interface Prototype for Assistive Robots for People Living with Upper Limb Disabilities," in *Rehabilitation Robotics (ICORR), 2019 International Conference on*, IEEE, 2019.

[60] C. L. Fall, A. Campeau-Lecours, C. Gosselin, and B. Gosselin, "Evaluation of a Wearable and Wireless Human-Computer Interface Combining Head Motion and sEMG for People with Upper-Body Disabilities," in *IEEE International NEWCAS Conference*, IEEE, 2018.

[61] C. L. Fall, U. Côté-Allard, Q. Mascret, A. Campeau-Lecours, C. Gosselin, and B. Gosselin, "Toward a Flexible and Modular Body-Machine Interface for Individuals Living with Severe Disabilities : a Feasibility Study," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 2019.

[62] U. Cote-Allard, C.-L. Fall, A. Campeau-Lecours, C. Gosselin, F. Laviolette, and B. Gosselin, "Deep Learning for Electromyographic Hand Gesture Signal Classification Using Transfer Learning," *IEEE Transactions on Biomedical Engineering*, 2019.

[63] F. Nougarou, A. Campeau-Lecours, D. Massicotte, and B. Gosselin, "Muscle Activity Distribution Features Extracted from HD sEMG to Perform Forearm Pattern Recognition," in *IEEE International NEWCAS Conference*, IEEE, 2018.

[64] C. L. Fall, G. Gagnon-Turcotte, J.-F. Dubé, J. S. Gagné, Y. Delisle, A. Campeau-Lecours, C. Gosselin, and B. Gosselin, "Wireless sEMG-based body–machine interface for assistive technology devices," *IEEE journal of biomedical and health informatics*, vol. 21, no. 4, pp. 967–977, 2017.

[65] C. L. Fall, P. Turgeon, A. Campeau-Lecours, V. Maheu, M. Boukadoum, S. Roy, D. Massicotte, C. Gosselin, and B. Gosselin, "Intuitive wireless control of a robotic arm for people living with an upper body disability," in *Engineering in Medicine and Biology Society (EMBC), 2015 37th Annual International Conference of the IEEE*, pp. 4399–4402, IEEE, 2015.

[66] R. Crepin, C. L. Fall, Q. Mascret, C. Gosselin, A. Campeau-Lecours, and B. Gosselin, "Real-Time Hand Motion Recognition Using sEMG Patterns Classification," in *2018 40th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pp. 2655–2658, IEEE, 2018.

[67] U. Côté-Allard, C. L. Fall, A. Drouin, A. Campeau-Lecours, C. Gosselin, K. Glette, F. Laviolette, and B. Gosselin, "Deep Learning for Electromyographic Hand Gesture Signal Classification by Leveraging Transfer Learning," *arXiv preprint arXiv :1801.07756*, 2018.

[68] C. L. Fall, F. Quevillon, M. Blouin, S. Latour, A. Campeau-Lecours, C. Gosselin, and B. Gosselin, "A multimodal adaptive wireless control interface for people with upper-body disabilities," *IEEE transactions on biomedical circuits and systems*, vol. 12, no. 3, pp. 564–575, 2018.

[69] A. Lebrasseur, J. Lettre, F. Routhier, P. Archambault, and A. Campeau-Lecours, "Assistive robotic arm : Evaluation of the performance of intelligent algorithms," *Assistive Technology Journal*, 2019.

[70] J. Schmidtler, K. Bengler, F. Dimeas, and A. Campeau-Lecours, "A questionnaire for the evaluation of physical assistive devices (QUEAD) : Testing usability and acceptance in physical human-robot interaction," in *Systems, Man, and Cybernetics (SMC), 2017 IEEE International Conference on*, pp. 876–881, IEEE, 2017.

[71] A. Lebrasseur, J. Lettre, F. Routhier, Archambault Philippe, and A. Campeau-Lecours, "Assistive robotic device : evaluation of intelligent algorithms," *Rehabilitation Engineering and Assistive Technology Society of North America (RESNA)*, 2018.