Commande robuste d'objets déformables avec des bras robotiques et application à un procédé industriel

Victor Giraud

THÈSE de DOCTORAT DE L'UNIVERSITÉ CLERMONT AUVERGNE

Ecole Doctorale Sciences pour l'Ingénieur

Institut Pascal - UMR 6602

Spécialité de doctorat : Image, Système de perception et Robotique

Soutenue publiquement le 24/01/2024, par :

Victor Henri Giraud

Commande robuste d'objets déformables avec des bras robotiques et application à un procédé industriel

---

Devant le jury composé de :

Yuri LAPUSTA Professeur des Universités, Clermont Auvergne INP	**Président du jury**
Jean-Pierre GAZEAU Ingénieur de recherche, HDR, Institut P Prime	**Rapporteur**
Omar TAHRI Professeur des Universités, Université de Bourgogne	**Rapporteur**
Juan Antonio CORRALES RAMON Post-Doctorant, Universidade de Santiago de Compostela	**Examinateur**
Mélodie DANIEL Maîtresse de Conférences, Université de Bordeaux	**Examinatrice**
Chedli BOUZGARROU Professeur des Universités, Clermont Auvergne INP	**Directeur de thèse**
Youcef MEZOUAR Professeur des Universités, Clermont Auvergne INP	**Co-Directeur de thèse**
Erol OZGUR Maître de Conférences, Clermont Auvergne INP	**Co-Encadrant**

Préface

Remerciements

Je voudrais exprimer avant toute chose mes remerciements :

A ma compagne, Julie, qui m'a supporté pendant la durée de cette thèse.

A l'Union Européenne et l'État Français pour avoir financé ces travaux.

A mes parents, Valérie et Marc, qui m'ont élevé, donné les moyens de nourrir ma curiosité et soutenu financièrement pour 5 ans d'études supérieures, et à mon grand-père Louis qui m'a toujours poussé à me dépasser, ainsi qu'aux autres membres de ma famille : Camille, Tiana, Quentin, Emilie, Henriette, Nicole, Christophe, Valérie, Pauline, Eliane, Annie et Jean.

A mon directeur de thèse, Chedli Bouzgarrou, pour sa disponibilité et ses grandes capacités d'écoute.

Aux autres membres de l'équipe d'encadrement, bien sûr du côté académique - Erol Ozgur, Youcef Mezouar, mais aussi du côté industriel Jean-Marie d'Ettorre pour sa grande humanité et sa capacité à tout faire pour donner les moyens à nos ambitions.

Aux nombreux camarades, que ce soit à Michelin (Nico et Nico, Fabrice, Michel), ou parmi les autres membres du consortium (Carlos, Juan Antonio, Daniel, Eric, Timo, Elias, Azad) ainsi qu'à l'Institut Pascal (Hernan, Miguel, Reza, Rohit, Omid, Maxime, Bastien, Jules, Samuel, Romain).

A mes amis qui m'ont soutenu tout au long de cette thèse : Kévin, sans qui je n'aurais pas eu ce poste, François, Rémi, Simon et Florence, Elodie et Johnatan, Mélodie et Rémy, Cécile, Pierre-Philippe.

A mes compagnons d'entraînement : mon maître Gérard, Suzanne, Luc, Alexis, Jean-Louis, Baptiste, Boris, Blaise, Ian, Christophe, David, Benoît, William, Kévin, Elodie, Madi, et tous les autres.

Acknowledgements

Ces travaux de recherche ont été financés par le laboratoire public-privé Factolab [1], dans le cadre du projet Européen SoftManBot.

1. http ://www.imobs3.uca.fr/index.php/fr/menu-valorisation/factolab

Abstract

Version Française

Les objets déformables sont omniprésents. Sous forme de câbles, vêtements, plastiques, ils font parti du quotidien. Ces objets doivent être manipulés, fabriqués, transportés. Leur déformabilité rend ces tâches plus ardues que pour leurs homologues rigides. Les travaux de cette thèse se focalisent sur la résolution d'un cas concret industriel, non résolu et applicativement intéressant : l'assemblage de bande de roulement de pneumatiques pour poids lourds, partie au contact de la route, procédé encore manuel actuellement. Ce procédé industriel est proposé par le partenaire industriel Michelin au sein du consortium SoftManBot, programme H2020 de l'Union Européenne ayant pour ambition d'automatiser la production industrielle d'objets déformables. La manipulation d'objets déformables soulève plusieurs problèmes que ne présentent pas les objets rigides : un problème de modélisation, un problème de perception, un problème d'asservissement de forme, et un problème d'ingénierie système pour faire fonctionner tous les composants précédents de concert.

Dans cette thèse, nous proposons deux contributions majeures. La première, Optimal Shape Servoing, est une commande par retour d'état basée sur la commande optimale améliorant l'état de l'art de l'asservissement de forme en ajoutant une gestion implicite de la trajectoire de déformation – la manière dont l'objet atteint la déformation finale. De plus, la stratégie de commande permet de découpler et pondérer les erreurs de forme et de position. Enfin, cette thèse expose un apprentissage par démonstration des paramètres du contrôleur grâce à un algorithme génétique pour suivre le comportement de l'objet manipulé par un humain, afin de reproduire cette déformation lors de tâches de manipulation. L'identification de ces paramètres grâce aux stratégies de Machine Learning permet le meilleur des deux mondes - à la fois un fonctionnement explicable et un comportement proche de l'opération effectuée par un humain.

Notre seconde contribution, Holistic Architecture for Deformable Object Software, répond au problème d'ingénierie système en proposant une architecture logicielle modulaire formalisant les besoins et interfaces nécessaires pour les problèmes de manipulation d'objets déformables, en laboratoire et en contexte industriel, de l'interface utilisateur au driver des préhenseurs. Cette architecture a été validée et testée par l'intégration de nombreuses briques logicielles - modèles, commandes, perception, interface utilisateur, contrôleurs de robots, drivers de caméra, drivers de préhenseurs. Ces briques sont comparées objectivement grâce aux métriques industrielles gouvernant la qualité d'un produit final, permettant non seulement d'automatiser la tâche proposée mais aussi de choisir la combinaison de modules la plus adaptée à cette même tâche.

Mots-clés : Robotique, Objet déformable, Architecture Logicielle, Commande Optimale, Asservissement de forme

English version

Deformable objects are ubiquitous. In the form of cables, clothing, plastics, they are part of everyday life. These objects need to be manipulated, manufactured, and transported. Their deformability makes these tasks more challenging than for their rigid counterparts. The work of this thesis focuses on solving a specific industrial case, which is unresolved and of practical interest : the assembly of heavy-duty tire treads, the part that comes into contact with the road, which is still a manual process. This industrial process is proposed by the industrial partner Michelin within the SoftManBot consortium, a Horizon20 program of the European Union with the ambition to automate the industrial production of deformable objects. The manipulation of deformable objects raises several problems that rigid objects do not present : a modeling problem, a perception problem, a shape servoing problem, and a system engineering problem to make all the preceding components work together.

In this thesis, we propose two major contributions. The first one, Optimal Shape Servoing, is a state feedback control based on optimal control that improves the state of the art in shape control by adding an implicit management of the deformation trajectory - how the object reaches its final deformation. Furthermore, the control strategy allows for decoupling and weighting shape and position errors. Finally, this thesis presents a demonstration-based learning of controller parameters using a genetic algorithm to mimic the behavior of an object manipulated by a human, in order to reproduce this deformation during manipulation tasks. Identifying these parameters through machine learning strategies combines the best of both worlds - both explainable operation and behavior close to that performed by a human.

Our second contribution, Holistic Architecture for Deformable Object Software, addresses the system engineering problem by proposing a modular software architecture that formalizes the needs and interfaces required for deformable object manipulation problems, both in the laboratory and in an industrial context, from user interface to gripper drivers. This architecture has been validated and tested through the integration of numerous software components - models, controls, perception, user interfaces, robot controllers, camera drivers, gripper drivers. These components are objectively compared using industrial metrics governing the quality of a final product, allowing not only the automation of the proposed task but also the selection of the most suitable combination of modules for the same task.

Keywords : Robotics, Deformable Object, Software Architecture, Optimal Control, Shape Servoing

Table des matières

Table des figures ix

Liste des tableaux xi

Liste des algorithmes xiii

Glossaire xv

Introduction 1

1 Présentation de la problématique industrielle et des bancs de test 5
1.1 Présentation du cas d'usage privilégié : l'assemblage de la bande de roulement . . . 5
1.1.1 A propos de la gomme . . . 5
1.1.2 Procédé typique d'assemblage de pneu . . . 7
1.1.3 Spécificités de l'assemblage de la bande de roulement . . . 8
1.2 Présentation des bancs de test . . . 10
1.2.1 Banc de test industriel . . . 13
1.2.2 Banc de test académique . . . 14
1.3 Méthodologie de développement logiciel . . . 15
1.4 Conclusion . . . 16

2 État de l'art 17
2.1 Modélisation de déformation . . . 18
2.1.1 Modèles physiques . . . 18
2.1.1.1 Approche de modèle masse-ressort . . . 18
2.1.1.2 Approche Dual-Quaternion . . . 19
2.1.1.3 Approche éléments finis . . . 20
2.1.1.4 Vue d'ensemble des modèles physiques . . . 21
2.1.2 Modèles géométriques . . . 21
2.1.2.1 Position Based Dynamics . . . 21
2.1.2.2 Meshless Shape Matching . . . 22
2.1.2.3 As-Rigid-As-Possible . . . 23
2.1.2.4 Vue d'ensemble des modèles géométriques . . . 23
2.1.3 Approches par apprentissage . . . 24
2.1.4 Approches sans modèle . . . 24
2.1.5 Vue d'ensemble des modèles . . . 24
2.1.6 Réduction de l'ordre du problème . . . 25
2.2 Méthodes de perception informatique . . . 25
2.2.1 Méthodes de perception analytique . . . 25
2.2.2 Méthodes de perception par réseaux de neurones . . . 27

2.2.2.1 Architectures des réseaux de neurones . . . 27
2.2.2.2 Techniques d'entraînement de Réseaux de Neurones . . . 28
2.2.2.3 Perception d'objets déformables avec des réseaux de neurones . 29
2.2.2.4 Contrôle de robots par réseaux de neurones . . . 30
2.3 Lois de commande pour l'asservissement de forme . . . 30
2.3.1 Contrôle-commande classique . . . 30
2.3.1.1 Commande PID . . . 31
2.3.1.2 Commande optimale . . . 31
2.3.1.3 Commande H_∞ . . . 32
2.3.1.4 Model Predictive Control . . . 32
2.3.1.5 Asservissement visuel . . . 33
2.3.1.6 Commande adaptative . . . 33
2.3.1.7 Stabilité . . . 33
2.3.1.8 Conclusion . . . 33
2.3.2 Asservissement de forme . . . 34
2.3.2.1 Asservissement de forme en infographie . . . 35
2.3.2.2 Asservissement de forme en robotique . . . 35
2.4 Architectures logicielles pour systèmes robotiques . . . 38

3 Contribution : Optimal Shape Servoing 41
3.1 Positionnement relativement à l'état de l'art . . . 41
3.2 Présentation de la stratégie Optimal Shape Servoing . . . 43
3.2.1 Synthèse des paramètres de commande . . . 44
3.3 Représentation algorithmique de la stratégie `OSS` . . . 46
3.4 Expériences et Résultats . . . 47
3.4.1 Tâche simulée . . . 47
3.4.2 Tâche d'assemblage de bandes de gomme . . . 49
3.4.2.1 Cas d'utilisation industriel . . . 49
3.4.2.2 Méthodologie expérimentale . . . 50
3.4.2.3 Résultats . . . 51
3.4.3 Mise en boîte d'un poster . . . 52
3.4.3.1 Description de la tâche . . . 52
3.4.3.2 Expériences . . . 53
3.4.3.3 Résultats . . . 55
3.4.4 Apprentissage par démonstration des paramètres des matrices de poids avec un algorithme génétique . . . 56
3.4.4.1 Description de la tâche . . . 56
3.4.4.2 Expériences . . . 56
3.4.4.3 Résultats . . . 60
3.5 Conclusion . . . 62

4 Proposition, intégration et performances d'`Hades`, architecture logicielle dédiée à la manipulation d'objets déformables 65
4.1 Formalisation du problème et analyse des besoins . . . 66
4.1.1 Besoins pour la manipulation robotique d'objets déformables . . . 67
4.1.2 Réutilisabilité . . . 70
4.1.3 Positionnement relatif à l'état de l'art . . . 71
4.2 Proposition d'architecture logicielle : `Hades` . . . 72
4.2.1 Structure haut niveau : tâches et skills . . . 72

4.2.2 Description d'`Hades` . . . 73
4.2.3 Interfaces . . . 76
4.3 Cas simplifié de validation . . . 80
4.3.1 Structure haut niveau . . . 81
4.3.2 Nœud supervisor . . . 81
4.3.3 Nœud perception . . . 81
4.3.4 Nœud deformationControl . . . 82
4.3.5 Nœud skilledWorkcell . . . 82
4.3.6 Conclusion . . . 82
4.4 Cas d'usage et d'intégration : banc de test académique . . . 82
4.4.1 Modules de Supervisor . . . 83
4.4.2 Modules de SkilledWorkcell . . . 84
4.4.3 Modules de Perception . . . 85
4.4.4 Modules de DeformationControl . . . 89
4.5 Cas d'usage et d'intégration : banc de test industriel . . . 93
4.5.1 Modules de Supervisor . . . 94
4.5.2 Modules de SkilledWorkcell . . . 94
4.5.3 Modules de Perception . . . 96
4.5.4 Modules de DeformationControl . . . 102
4.5.5 Conclusion . . . 105
4.6 Comparaison des modules développés . . . 105
4.6.1 Comparaison des méthodes de perception . . . 106
4.6.2 Comparaison des stratégies de contrôle en boucle ouverte . . . 107
4.6.3 Comparaison des stratégies de contrôle en boucle fermée . . . 107
4.7 Discussion . . . 108

Conclusion **111**

A Préoccupations éthiques **115**

B Stratégie de saisie de la gomme **119**

C Approches prometteuses et infructueuses **123**

D Bibliographie **125**

Table des figures

1.1 Vue de la gomme crue extrudée faiblement déformée . . . 6
1.2 Erreur d'alignement initial de la gomme . . . 9
1.3 Représentation schématique de la tâche . . . 10
1.4 CAD du banc de test industriel . . . 11
1.5 Illustration labellisée du banc de test industriel . . . 12
1.6 Synoptique de sécurité . . . 13
1.7 Banc de test académique . . . 15

2.1 Illustration de modèle masse-ressort et son application . . . 19
2.2 Illustration de modèle élément finis . . . 20
2.3 Illustrations générées avec Position Based Dynamics . . . 21
2.4 Illustrations générées avec Meshless Shape Matching . . . 22
2.5 Objet complexe dont les déformations sont simulées avec ARAP . . . 23
2.6 Illustration du fonctionnement de l'algorithme Shape-from-Template . . . 26
2.7 Architecture du réseau UNet . . . 28
2.8 Simulation de la mise en place d'une nappe . . . 34
2.9 Simulation sous contrainte de l'animation d'un personnage en forme de X . . . 35
2.10 Asservissement de la tension d'une bande de gomme grâce à l'équation du caténaire 36
2.11 Asservissement d'un objet volumique . . . 37

3.1 Schéma bloc d'OSS. . . . 44
3.2 Représentation sous forme de "Carte thermique" des valeurs de la matrice Q sur les nœuds du maillage. . . . 45
3.3 Convergence de forme ARAP-SS et OSS au cours de l'exécution . . . 48
3.4 Erreur d'asservissement en fonction du temps . . . 49
3.5 OSS scheme servoing error . . . 49
3.6 Assemblage manuel de la gomme comme fait en usine. . . . 50
3.7 Maillage perçu de la bande de roulement . . . 51
3.8 Comparaison des erreurs de trajectoire . . . 51
3.9 Biais imposant une déformation à une tâche de translation . . . 52
3.10 Configuration de la tâche de mise en boîte d'un poster . . . 53
3.11 Forme cible pour la mise en boîte . . . 54
3.12 Vue du dessus de la tâche à l'instant critique . . . 54
3.13 Erreur de forme en fonction de l'erreur de position . . . 55
3.14 Déformation cible aux instants 1, 25, 50, 75 et 100 . . . 56
3.15 Valeurs indentifiées des coefficients de la matrice Q pour l'objet linéique . . . 57
3.16 Déformation aux instants 1, 25, 50, 75 et 100 . . . 60
3.17 Commandes générées par stratégie d'asservissement de forme . . . 61
3.18 Echec de convergence des contrôleurs, instants 51, 75 et 100 . . . 61

3.19 Commandes générées avec un bruit blanc gaussien de perception 62
3.20 Erreur de trajectoire des différentes stratégies de contrôle 62

4.1 Représentation modulaire de l'architecture matérielle et logicielle générique . . . 68
4.2 Description de la séquence d'exécution souhaitée par un système logiciel de manipulation d'objet déformable 69
4.3 Schéma bloc d'`Hades` 74
4.4 Interface entre parties génériques et spécifiques 77
4.5 Graph ROS d'`Hades` en fonctionnement 78
4.6 Interface Homme Machine utilisée pour les cas d'usages 79
4.7 Vue de la tâche robotique lors du cas de validation 80
4.8 Configuration du cas de validation 81
4.9 Configuration du banc de test académique 83
4.10 Header des robots 85
4.11 Header des grippers 86
4.12 Header de camera 86
4.13 Header de la perception ROBUSfT 87
4.14 Header de la perception LARAP 88
4.15 Objet déformable utilisé pour les expériences 89
4.16 Résultat de la stratégie appliquée à un seul manipulateur pour une tâche de Pick and Place 90
4.17 Interface de l'implémentation du module de modèle ARAP 91
4.18 Interface de l'implémentation du contrôleur jacobien-proportionnel 91
4.19 Interface de l'implémentation du contrôleur Optimal Shape Servoing 92
4.20 Configuration du banc de test industriel 93
4.21 Header de la communication avec le PLC 95
4.22 Tread texturé pour les tests de ROBUSfT avec controlleur `OSS` sur le banc de test Michelin 97
4.23 Visualisation du suivi de forme LARAP pour la tâche en cours 98
4.24 Interface de la détection de caractéristiques 99
4.25 Résultat de la détection de caractéristiques 99
4.26 Procédé d'entraînement d'UNet 100
4.27 Résultat en direct de la perception basée sur UNet 100
4.28 Interface de la perception UNet 101
4.29 Procédé d'apprentissage de trajectoire par démonstration 103
4.30 Erreurs indépendantes de contrôle du contrôleur spécifique 103
4.31 Obtention des erreurs du contrôleur spécifique 104
4.32 Interface du contrôle spécifique 104
4.33 Bande de roulement avec marqueurs ArUco pour valider la stratégie proportionnelle spécifique 105
4.34 Visualisation du setup 106
4.35 Illustration du biais de sur-estimation dûe aux dimensions du système 112

B.1 Décomposition de la trajectoire de saisie 119
B.2 Adaptation de la trajectoire de saisie grâce aux inputs de perception 120
B.3 Design de quelques itérations des doigts 120
B.4 Impact de la forme des doigts sur l'intégrité de la gomme 121

C.1 Vue de la tentative de labellisation des points d'intérêts 123
C.2 Résultat de la méthode Graph Neural Network 124

Liste des tableaux

2.1 Table de comparaison des modèles physiques 21
2.2 Table de comparaison des modèles géométriques 23
2.3 Table de comparaison des modèles 25
2.4 Tableau comparatif des perceptions analytiques 27
2.5 Tableau récapitulatif des architectures de réseaux de neurones existantes 28
2.6 Tableau récapitulatif des techniques d'apprentissage existantes 29
2.7 Table de comparaison des lois de commande 34

3.1 Comparaison de l'état de l'art des stratégies dédiées à la manipulation d'objets déformables 42

4.1 Solutions de réutilisabilité pour exécutables logiciels 71
4.2 Table de comparaison de l'état de l'art des architectures 72
4.3 Précision par pixel par résolution à hauteur de gomme dans le cas d'usage 106
4.4 Performance des méthodes de perception 106
4.5 Récapitulatif des méthodes de perception appliquées à la tâche 107
4.6 Récapitulatif des stratégies boucle ouverte appliquées à la tâche 107
4.7 Récapitulatif des stratégies boucle fermée appliquées à la tâche 108

Liste des algorithmes

1 Stratégie `OSS` 47
2 Création automatique de la matrice Q 55
3 Génération de la forme cible 57
4 Algorithme génétique proposé 58
5 Fonctions de mutation 59
6 Machine d'état 75
7 Implémentation de la machine d'état piloté par l'IHM, côté IHM 84
8 Implémentation de la machine d'état piloté par l'IHM côté superviseur 84
9 Implémentation de la machine d'état piloté par le PLC 94
10 Exemple de tâche pilotée par PLC : Init 94
11 Implémentation de la détection UNet 101

Glossaire

ARAP As-Rigid-As-Possible. 23
ARAP-SS As-Rigid-As-Possible Shape Servoing. 111
CAD Computer Aided Design. 11
cf Confer. 32
CNN Convolutional Neural Network. 27
DDL Degré De Liberté. 18
DNN Dense Neural Network. 27
DQ Dual Quaternions. 19
FEM Finite Element Method. 20
GNN Graph Neural Network. 28
IBVS Image Based Visual Servoing. 33
IHM Interface Homme Machine. 68
LARAP Latticed As-Rigid-As-Possible. 87
LQ Linear Quadratic. 31
LQG Linear Quadratic Gaussian. 31
MPC Model Predictive Control. 32
MSM Meshless Shape Matching. 22
OSS Optimal Shape Servoing. 3
PBVS Position Based Visual Servoing. 33
PID Proportionnel Intégral Dérivé. 31
PLA Polyactide. 121
PLC Programmable Logic Controller. 14
RAII Ressource Acquisition Is Initialization. 15
ROS Robot Operating System. 38
SfT Shape-from-Template. 25

Introduction

Nous vivons actuellement la quatrième révolution industrielle. Après l'avènement des usines, propulsées par la vapeur et les débuts de la mécanisation ; après la prolifération de l'acier et de l'électricité, qui a permis la création des chaînes de montage ; après l'émergence des plastiques, de l'électronique et de la mise en réseau informatique du siècle dernier, le paradigme change une fois de plus. Les progrès dans la miniaturisation permettent aux petits systèmes d'intégrer la puissance de calcul et la connectivité réseau là où des serveurs centraux accessibles uniquement par un terminal étaient nécessaires il y a cinquante ans, ouvrant ainsi la voie à l'Internet des objets. L'explosion de la connectivité génère quotidiennement des exaoctets de données, qui sont filtrés à l'aide de techniques issues du Big Data. L'intelligence artificielle - en particulier, l'apprentissage automatique, exploitant le flot de données actuel - nous offre de nouveaux outils pour résoudre des problèmes qui étaient autrefois considérés comme impossibles à résoudre par des machines, tels que le diagnostic médical, le traitement du langage naturel ou la reconnaissance d'images. La fabrication additive ouvre de nouvelles voies pour la création de produits et réduit considérablement les coûts et le temps de prototypage matériel. La robotisation des tâches complexes s'attaque à des problèmes qui résistaient à ses efforts lorsqu'elle a été initialement introduite dans les usines, comme la manipulation d'objets déformables.

Le premier robot industriel déployé était Unimate, en 1961. Unimate manipulait des pièces moulées à haute température pour l'industrie automobile, un processus dangereux pour les travailleurs humains en raison de la température élevée et des fumées toxiques. Ce prototype a ouvert la voie à la robotique moderne, en mettant en valeur les nombreux avantages de l'utilisation d'un robot pour une tâche industrielle. Premièrement, la productivité. Le robot peut être plus efficace qu'un être humain, ce qui permet de réduire les coûts. Deuxièmement, l'assurance qualité. Les mouvements des robots sont hautement répétables, ils ne deviennent pas moins précis à mesure que la fatigue s'accumule, et ils peuvent être facilement reproduits sur une autre machine. Troisièmement, la sécurité. La robotisation permet d'éviter d'exposer les travailleurs à des conditions ou des produits chimiques dangereux. Quatrièmement, le manque de main-d'œuvre. Les emplois d'opérateur d'usine ne sont pas attrayants, et de nombreuses industries peinent à trouver du personnel pour leurs chaînes de production, notamment dans l'Union européenne.

Les objets déformables sont partout : le caoutchouc, les plastiques, les cordes, les câbles, les textiles. Ces objets sont fabriqués, mais toutes les tâches associées ne sont pas encore automatisées. Si un objet n'est plus rigide, plusieurs problèmes se posent. La détection de la prise ne peut plus être simplifiée en mesurant le retour de force du préhenseur. La forme de notre objet déformable doit être connue. Imposer un changement de forme nécessite un modèle, et atteindre une forme désirée n'est pas une tâche triviale. Par exemple, est-il même possible d'atteindre la forme désirée au départ ?

En plus des applications industrielles, les techniques de manipulation des objets déformables peuvent également être appliquées aux aliments et aux tissus biologiques. Ces techniques ouvrent donc des portes à l'amélioration des processus de transformation alimentaire et à l'automatisation de la cuisine, ainsi que l'amélioration des pratiques médicales. Ainsi, la manipulation robotique des objets déformables est un problème ouvert d'un intérêt majeur.

Présentation du projet SoftManBot

SoftManBot [2], SOFT object MANipulation using roBOTs, est un projet européen de recherche du programme H2020. Le but du projet est la création de nouvelles solutions robotiques pour la production industrielle d'objets nécessitant des manipulations d'objets déformables. Le but est d'automatiser les tâches à faible valeur ajoutée afin de permettre la relocalisation de ces productions en Europe. Les acteurs du projet sont un consortium de onze institutions. Les membres sont des universités (Clermont Auvergne INP et Sorbonne Université), des instituts de recherche technologique (AIJU, INESCOP, l'Institut Italien de Technologie), des entreprises Business-to-Business intéressées par le développement de solution (STAM et Zimmer), et enfin des entreprises Business-to-Customer qui produisent et vendent l'objet déformable (Michelin, Plastinher, Juema et Decathlon). Les partenaires industriels apportent le cas d'usage réel et le problème à résoudre, ainsi que moyens humains et matériels de l'industrie pendant que les partenaires académiques apportent le savoir-faire théorique de pointe et la capacité de résolution de problèmes de recherche.

Dans le cas d'usage de Michelin, l'objet déformable produit est un pneu de poids lourd. Le but, dans le contexte du projet, est d'automatiser la pose de la bande de roulement, partie en gomme au contact de la route. La bande de roulement, aussi appelée KM, est la dernière partie du pneu posée pendant la phase d'assemblage, avant le procédé de cuisson permettant la vulcanisation. A priori, le défi de l'automatisation de cette tâche est amplifié par deux propriétés de la bande de roulement crue. Premièrement, l'épaisseur du produit implique qu'aucune dimension spatiale (largeur, longueur, hauteur) ne peut être trivialement simplifiée. Deuxièmement, la géométrie du produit, sillonné, rend la flexion autour de l'axe longitudinal plus facile que la flexion autour de l'axe latéral : le produit est anisotrope.

Dans le cas d'usage de Plastinher, l'objet déformable produit est une chaussure. Le but est d'automatiser l'extraction de semelle. Cette tâche est physiquement exigeante, consistant à retirer les semelles d'un moule chaud avec des pinces. L'industrie de la mode exigeant une capacité d'adaptation rapide à de nouveaux modèles, la propriété la plus importante d'une solution est son adaptabilité à des nouveaux moules ou modèles de semelle.

Dans le cas d'usage de Juema, l'objet déformable produit est une poupée. Le but est d'automatiser deux procédés : l'extraction de pièces moulées, et l'assemblage de la poupée. Ces deux procédés nécessitent que les pièces soient encore chaudes, donc plus déformables que leurs versions froides. Cette contrainte rend la vitesse d'exécution des solutions cruciale.

Dans le cas d'usage de Decathlon, l'objet déformable produit est un short de cycliste. Le but est de coudre un rembourrage en mousse sur le tissu élastique tout en évitant les plis. Il faut donc pouvoir détecter, et éliminer ces plis avant de placer le rembourrage et de commencer la couture. La difficulté principale de la tâche est la transition d'une forme 2D, lors de la couture du rembourrage, à une forme 3D sans plis, lors de la couture de la taille et des jambes.

Les travaux de cette thèse sont issus de la volonté de résoudre le cas d'usage Michelin, puis le généraliser. Les autres cas d'usages présentés sont autant de scénarios de généralisation et de transfert possible de nos développements. En temps que membres d'un même projet, nous avons partagé codes sources, techniques, et surtout nos idées et procédés provenant de nos expériences et industries respectives, pour atteindre notre but commun d'automatisation de production d'objets déformables.

2. https ://www.softmanbot.eu/

Contributions

Cette thèse apporte deux contributions majeures.

- La première contribution est la proposition d'une stratégie de contrôle, Optimal Shape Servoing (`OSS`), permettant un asservissement de forme d'un objet déformable en suivant une trajectoire de déformation, c'est à dire le contrôle des formes intermédiaires prises par l'objet pendant l'asservissement. Nous détaillerons cette méthodologie et son application à plusieurs cas de simulation et cas réels dans le chapitre 3. Cette contribution est formée
  1. de la stratégie d'asservissement elle-même
  2. de son application pour dé-corréler les erreurs de forme et de position
  3. de l'apprentissage par démonstration des paramètres de la commande
- La seconde contribution est la proposition d'`Hades`, Holistic Architecture for DEformable objects Software, une architecture logicielle pour la manipulation d'objets déformables. Nous détaillerons cette architecture dans le chapitre 4. Cette contribution est formée
  1. de l'architecture `Hades` elle-même
  2. de l'adaptation des divers développements du projet pour les intégrer sous forme de modules dans l'architecture
  3. de l'analyse comparative des différentes techniques que propose l'état de l'art en terme de perception et de contrôle d'objets déformables

Liste des résultats valorisés

Journaux et conférences académiques :

— Premier auteur de "Optimal Shape Servoing", accepté pour IROS 2022 [Gir+22]
— Second auteur de "Dynamic movement primitive using Dual Quaternions," accepté pour ICRA 2023 [Cha+23]

Brevet industriel :

— Brevet soumis [Gir+23]. ID temporaire : 2022PAF000221FR. Validation finale attendue pour Septembre 2024

Plan de la thèse

Le manuscrit de thèse est organisé de la manière suivante :

Le chapitre 1 détaille le procédé complet de la fabrication d'un pneu poids lourd, des matériaux au produit final. Ce point de départ est l'élément motivant nos recherches, et permet de contextualiser la tâche à automatiser. Cette dernière est décrite, avec le détail des défis à relever et décrivant les conditions de succès de la tâche. Enfin, nous présentons les bancs de test utilisés pour les développements présentées dans cette thèse.

Le chapitre 2 détaille l'état de l'art, organisé selon les champs thématiques abordés dans cette thèse. Le premier point étudié est la modélisation de déformation, c'est à dire la manière de calculer la forme attendue d'un objet déformable soumis à des contraintes de déplacement ou de force connus. Nous étudions ensuite les stratégies de commandes classiques, et leurs applications potentielles au cas des objets déformables. Nous étudions ensuite les méthodes de vision par ordinateur

appliquées à la détection de la forme d'un objet déformable, que ce soit par des méthodes analytiques ou par l'utilisation de réseaux de neurones. Par la suite, nous étudions les stratégies de l'état de l'art d'asservissement de forme. Enfin, nous passerons en revue les architectures logicielles adaptées au cas de la manipulation robotique d'objets déformables.

Le chapitre 3 présente une stratégie novatrice que nous proposons pour l'asservissement de forme : Optimal Shape Servoing (`OSS`). Elle prend en compte la trajectoire de déformation : la manière dont l'objet est déformé pour atteindre sa forme cible. Cette stratégie basée sur la commande optimale et la représentation d'état donne un sens physique aux paramètres des matrices de poids. Cette stratégie ouvre également des possibilités de dé-corréler erreur de forme et erreur de position. Nous avons également appliqué des techniques issues du Machine Learning pour apprendre les paramètres des matrices de poids par démonstration, et se rapprocher du geste effectué par un opérateur humain.

Le chapitre 4 présente Holistic Architecture for DEformable objects Software (`Hades`), une architecture logicielle dédiée au prototypage et au transfert à des applications industrielles de manipulation robotique d'objets déformables. En se basant sur les problèmes à résoudre dans le contexte du projet, nous proposons une architecture générique. Cette proposition comporte le détail de son fonctionnement interne, et de son application. Après un exemple simple sur un cas de validation, nous détaillons l'intégration dans `Hades` de travaux académiques et leur transfert en industrie. `Hades` a permis la comparaison objective des performances à l'échelle du système des différentes méthodes de perception et stratégies de contrôle appliquées à notre cas d'usage, que nous présentons donc.

Chapitre 1

Présentation de la problématique industrielle et des bancs de test

Comme ce qui a été introduit, notre cas d'usage privilégié dans le contexte du projet SoftManBot est l'assemblage de bande de roulement de poids lourds. Il s'agit d'une seule étape parmi les nombreuses nécessaires à la fabrication du pneu. Le procédé complet de fabrication de pneu est extrêmement délicat et complexe, à l'origine de dizaines de milliers de brevets industriels. La confection de pneumatiques implique des réactions chimiques complexes d'élastomère, des polymères élastiques, ainsi qu'une complexité mécanique insoupçonnée. L'aspect mécanique vient aussi bien du pneu lui-même, assemblage composite de nombreuses couches aux propriétés souhaitées bien définies, que des machines nécessaires à sa confection. La multitude de composants interne interagissent les uns avec les autres pour toute la durée de vie du produit, qui se trouve grandement amputée en cas de défaut d'un composant. Les machines de fabrication doivent donc manipuler avec le plus grand soin les parties délicates, pour permettre l'intégrité du pneu. L'enjeu final reste la sécurité des utilisateurs finaux : le pneu est un organe essentiel au bon fonctionnement d'une voiture, que ce soit pour maintenir l'adhérence sous des conditions climatiques difficiles, garantir la tenue de route et maintenir des faibles distances d'arrêt en cas de freinage d'urgence.

Ce chapitre résume le procédé de fabrication de pneu, en détaillant l'étape de l'assemblage de la bande de roulement. Ensuite, nous présentons les bancs de tests utilisés pour résoudre le cas d'usage privilégié.

1.1 Présentation du cas d'usage privilégié : l'assemblage de la bande de roulement

La soudure de la bande de roulement n'est qu'une étape du procédé complet. Les étapes précédentes et suivantes vont définir ce qui est ou n'est pas faisable pour la tâche à effectuer. Il faut donc avoir une vision d'ensemble de la fabrication d'un pneu pour que les travaux effectués aient tout leur sens.

1.1.1 A propos de la gomme

Les gommes sont des matériaux techniques fragiles. Un premier atelier mélange les matières premières pour obtenir un pain de gomme. Les matières premières en question sont :

— Le caoutchouc naturel est un matériau résistant et imperméable, capable de s'étirer considérablement. Il est obtenu à partir du latex de l'arbre nommé Hévéa. Le latex est une substance visqueuse de couleur blanche, qui n'est pas la sève de l'arbre : le latex est une partie

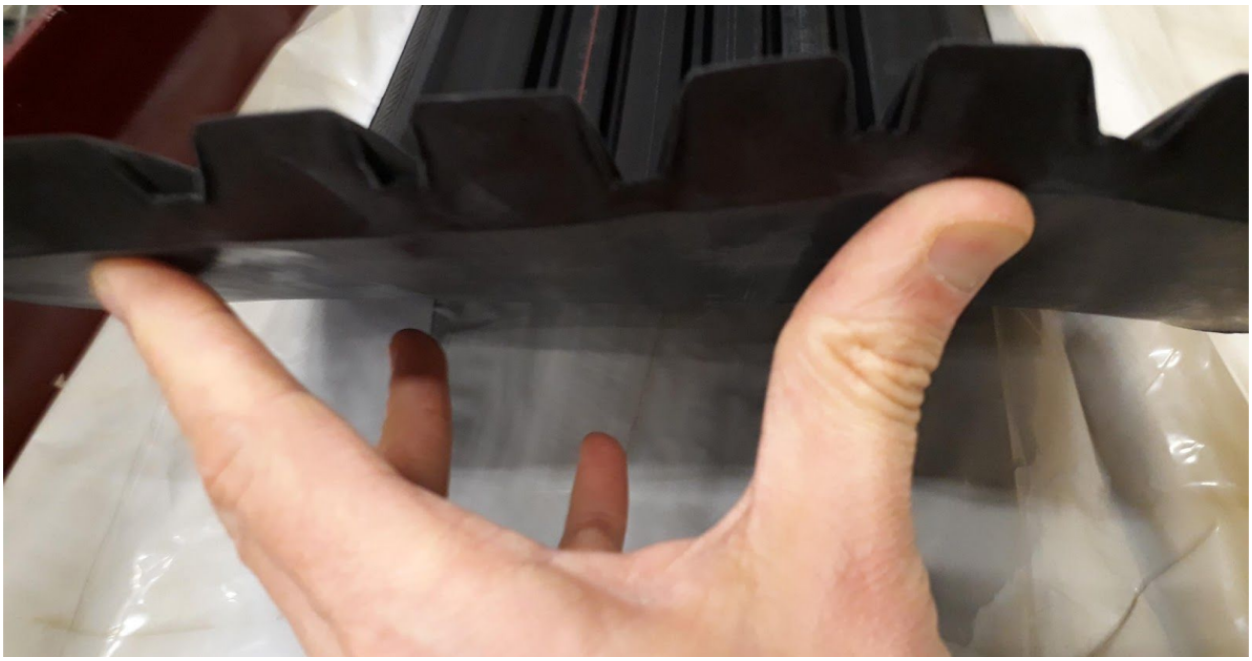

FIGURE 1.1: Vue de la gomme crue extrudée faiblement déformée

du mécanisme de défense de l'arbre contre les herbivores, car il contient des toxines et des biopesticides, et a la capacité de former une couche protectrice en cas de blessure potentielle de l'écorce. Pour obtenir le latex, le tronc de l'arbre est incisé à intervalle régulier. Ensuite, il est stabilisé en réduisant sa teneur en eau et en ajoutant de l'ammoniaque.

— Les élastomères, dérivés d'hydrocarbure, des polymères synthétiques visco-élastiques hautement déformables. La composition chimique exacte de l'élastomère déterminera les propriétés d'interface de la gomme, ainsi que sa résistance au roulement et la transmission de traction.
— Le noir de carbone, ce qui donne la couleur noire au pneu ainsi que la résistance à l'abrasion. Le noir de carbone est une forme de carbone paracrystallin, une poudre faite de sphères de taille nanométrique, obtenues en résultat d'une combustion incomplète de gaz naturel ou bien d'hydrocarbures.
— Les antioxydants et les antiozonants, pour diminuer les effets de l'exposition à l'oxygène, à l'ozone et à la chaleur
— La Silice, améliorant la résistance au roulement et donc la durée de vie du pneu.
— Le Soufre accélérant la vulcanisation, le procédé où les élastomères chauffés tissent des ponts entre les chaînes de polymère, sans direction privilégiée, rendant la gomme bien moins plastique et bien plus élastique.

La composition exacte de l'élastomère et les proportions d'additifs définiront les propriétés de la gomme stabilisée. Ainsi, des gommes avec des configurations différentes seront utilisées pour créer des gommes adaptée à la protection des pièces métalliques, ou adaptée à l'adhésion au tissu, ou encore adaptée à l'interface gomme-gomme, ou enfin adaptée à l'interface pneu-route.

Les palettes de pains de gomme sont envoyées aux ateliers préparatoire des usines. Ces ateliers remélangent les gommes pour préparer des produits adaptés aux besoins de l'usine. La gomme est extrudée, embobinée, envoyée aux divers postes de la chaîne, résultant en un produit montré dans la Fig. 1.1. Au poste de travail, les diverses gommes nécessaires sont déroulées, positionnées, coupées, soudées par le travailleur. La gomme crue se comporte comme du chewing-gum, avec des propriétés élastiques et plastiques. Une fois vulcanisée, la gomme perd ses propriétés plastiques, renforçant drastiquement les propriétés de rigidité et d'élasticité. La sensibilité du produit

cru génère des contraintes. Premièrement, le bois est strictement interdit en usine à proximité de la gomme crue : les échardes sont indétectables dans les procédés de vérification de qualité, et détruisent les pneus de l'intérieur. Les gommes imperméables ne doivent pas être mélangées avec les autres gommes, étant donné que les composants créant l'imperméabilité dégradent fortement les autres performances, même à des quantités infimes. Les gommes imperméables ont donc des machines dédiées et des chaînes de production séparées. La gomme crue doit être manipulée avec des gants, pour protéger le produit et le travailleur. Les huiles naturelles présentes dans la sueur et les peaux mortes dégradent le collant de la gomme, et les composés chimiques toxiques de la gomme non stabilisée peuvent rentrer dans le corps par le contact avec la peau. Enfin, des solvants volatiles sont nécessaires pour atteindre des niveaux d'adhésion suffisant entre les différentes couches de gomme du pneu.

1.1.2 Procédé typique d'assemblage de pneu

Le procédé de fabrication de pneu peut être décomposé en trois phases : Premièrement, la création de la carcasse. Deuxièmement, la conformation et l'assemblage du bloc sommet. Troisièmement, la cuisson.

Création de la carcasse : La carcasse est un châssis cylindrique autour duquel le pneu est construit. Elle est composée des couches suivantes :

- La première couche est faite de gomme imperméable, nécessaire car la plupart des pneus construits actuellement sont "tubeless" - sans chambre à air, le boyau de caoutchouc rempli d'air trouvable sur les vélos. Cette couche sera à même de retenir l'air sous pression entre le pneu et la roue.
- Ensuite, une "nappe zéro" est posée sur cette première couche. Cette nappe est faite de câbles d'aciers intégrés dans de la gomme. Le terme de "zéro" provient de l'angle entre les câbles de cette nappe et l'axe de rotation du pneu. Ces câbles sont donc orthogonaux à la route.
- D'autres couches de gomme sont assemblées
- Deux tringles circulaires sont posées sur la gomme
- Les couches précédentes sont repliées pour couvrir les tringles
- D'autres couches de gomme sont posées sur la carcasse, y compris les futurs flancs du pneu

Conformation et assemblage du bloc sommet : Lors de la seconde étape, la carcasse est conformée. En utilisant l'énergie pneumatique, le cylindre épais prend la forme d'un tore étroit, reconnaissable comme pneu. Conformer le pneu forme les flancs, permettant l'empilement des autres produits sur la carcasse. Les produits posés forment ce qu'on appelle le bloc sommet.

- La première couche est faite de gomme.
- Ensuite, une nappe sommet dont les angles des câbles métalliques sont autour de 30°.
- D'autres couches de gomme, protégeant le métal, sont empilées.
- Une autre nappe sommée avec un angle de -30° est empilée. Les divers câbles créent une structure de triangles déformables, la base de l'architecture du pneu radial.
- D'autres couches de gomme sont empilées.
- La carcasse est entourée par un "fil zéro", orthogonal à l'axe de rotation du pneu.
- D'autres couches de gomme sont empilées, incluant des nappes textiles.
- La bande de roulement est empilée, soudée, rouletée. C'est l'étape automatisée dans cette thèse.
- Les flancs sont repliés.

Cuisson : A ce stade, le pneu cru est fini. La prochaine étape est la cuisson. Le pneu est mis dans un grand moule chauffé, dont la forme créera les sculptures. La bande de roulement est déjà sillonnée depuis l'étape de l'extrusion, mais le procédé de cuisson imprimera les dessins spécifiques améliorant l'adhérance ou l'évacuation d'eau. C'est également là où la marque et les informations du pneu sont imprimées sur les flancs. Dans le moule, le souffre a réagi avec les élastomères dans un procédé irréversible, la vulcanisation. Les élastomères forment de multiples liens entre les chaînes de polymère, en utilisant le souffre comme pont. Le pneu possède dès lors toutes les propriétés mécaniques que nous lui connaissons à ce stade, et est prêt à être vendu après un contrôle qualité individuel.

1.1.3 Spécificités de l'assemblage de la bande de roulement

Des travaux de recherche relatifs aux nappes sommet avec des bras robotiques ont déjà été conduits. Ils ne s'appliquent malheureusement pas à la bande de roulement étant donné les différences du produit, qui nécessitent des traitements spécifiques.

Configuration de la tâche : La bande de roulement vient habituellement s'accrocher au bloc sommet sous le cylindre. L'adhésion de gomme à gomme est forte, nous pouvons postuler que cette partie de la bande de roulement est bien fixée. De plus, la répétabilité des machines est telle que l'ancrage est effectué au même endroit, à une précision en deçà du dixième de millimètre. Si ce n'est pas le cas, un technicien de maintenance réglera la machine pour résoudre le problème. Le cylindre tourne. Un outil spécialisé, une lame chauffée vibrant à haute fréquence, coupe la bande de roulement avec un angle aigu à une longueur correspondant au périmètre total du pneu. Le cylindre poursuit sa rotation et met le produit avec la zone à souder face à l'opérateur. Les deux pans de la bande de roulement sont orientés comme montré dans la Fig. 1.2 : avec un léger défaut d'alignement. L'opérateur saisit la bande de roulement, la décroche en la tirant vers lui, l'aligne contre la bande fixe en se servant des repères visuels des sillons et du grain d'orge, sur-épaisseur millimétrique située au milieu de la bande. Enfin, l'opérateur applique un effort en compression sur la gomme. Toutes ces étapes forment l'opération de soudure, schématisée dans la Fig. 1.3. Enfin, le cylindre tourne avec un outil spécial pressé contre la gomme. Cette opération s'appelle le rouletage. Le pneu est prêt à être envoyé en cuisson.

Spécificités de la saisie : Le produit manipulé définit les trajectoires d'approche possibles pour la saisie.

- Le produit ne comporte pas de câbles métalliques, excluant une saisie magnétique.
- Le produit est trop lourd pour des préhenseurs ventouses.
- Des préhenseurs-aiguilles risquent d'endommager le produit

D'où la décision pratique de travailler avec des préhenseurs de type pince.

Définition des métriques qualifiant l'assemblage : Une soudure est dite bonne si l'erreur en position des coordonnées x et z est inférieure à deux millimètres - aussi proche de 0 que possible. Suivant l'axe y, dans le sens des sillons, le biseau de la gomme ouvre les tolérances entre 0 et 3 millimètres - en pratique, on vise un recouvrement d'1 millimètre. De plus, l'intégrité structurelle de la bande de roulement doit être préservée. Le cycle inclut la saisie, l'alignement et la soudure des ailes. Ce cycle de soudure doit être effectué sous la barre des 45 secondes, idéalement sous la barre des 30 secondes. La répétabilité doit être de 95%, avec la possibilité de détecter les soudures incorrectes et soit de réessayer, soit de notifier un opérateur.

FIGURE 1.2: Erreur d'alignement initial de la gomme

Gauche : configuration initiale typique. Droite : configuration cible.

Dans cette thèse, nous nous intéressons au problème de l'alignement. Des solutions d'ingénierie ont été mises en place pour résoudre le problème de saisie, détaillées dans l'annexe B et un système annexe a été développé pour effectuer la soudure des ailettes.

Difficultés inhérentes de la tâche :

- Notre objet déformable est rigide ; contrairement à – par exemple une serviette, notre bande de roulement en gomme possède une résistance en compression et en traction. La tâche nécessite un couple suffisant pour être achevée.
- La gomme fraîche est collante. D'un côté, cela permet à la soudure de résister à l'ouverture des doigts. D'un autre côté, cela signifie que soulever la gomme collée et ajuster la bande de roulement en fin de tâche sont des défis supplémentaires.
- La bande de roulement est lourde. Un humain effectuant la tâche utilise ses deux mains et le couple estimé est proche de la limite possible des robots choisis, décrit dans la section 1.2.
- Les ailettes, formées d'une très faible épaisseur de gomme située sur les deux côtés de la bande de roulement, sont fragiles et doivent être manipulées avec délicatesse. Comme elles sont collantes, un doigt humain ou robotique entrant en contact avec une ailette la retournera et la collera sur elle-même, créant à la fois un trou et une sur-épaisseur observable dans la Fig. B.4, ce qui résulte en un produit final non conforme aux normes de qualité.

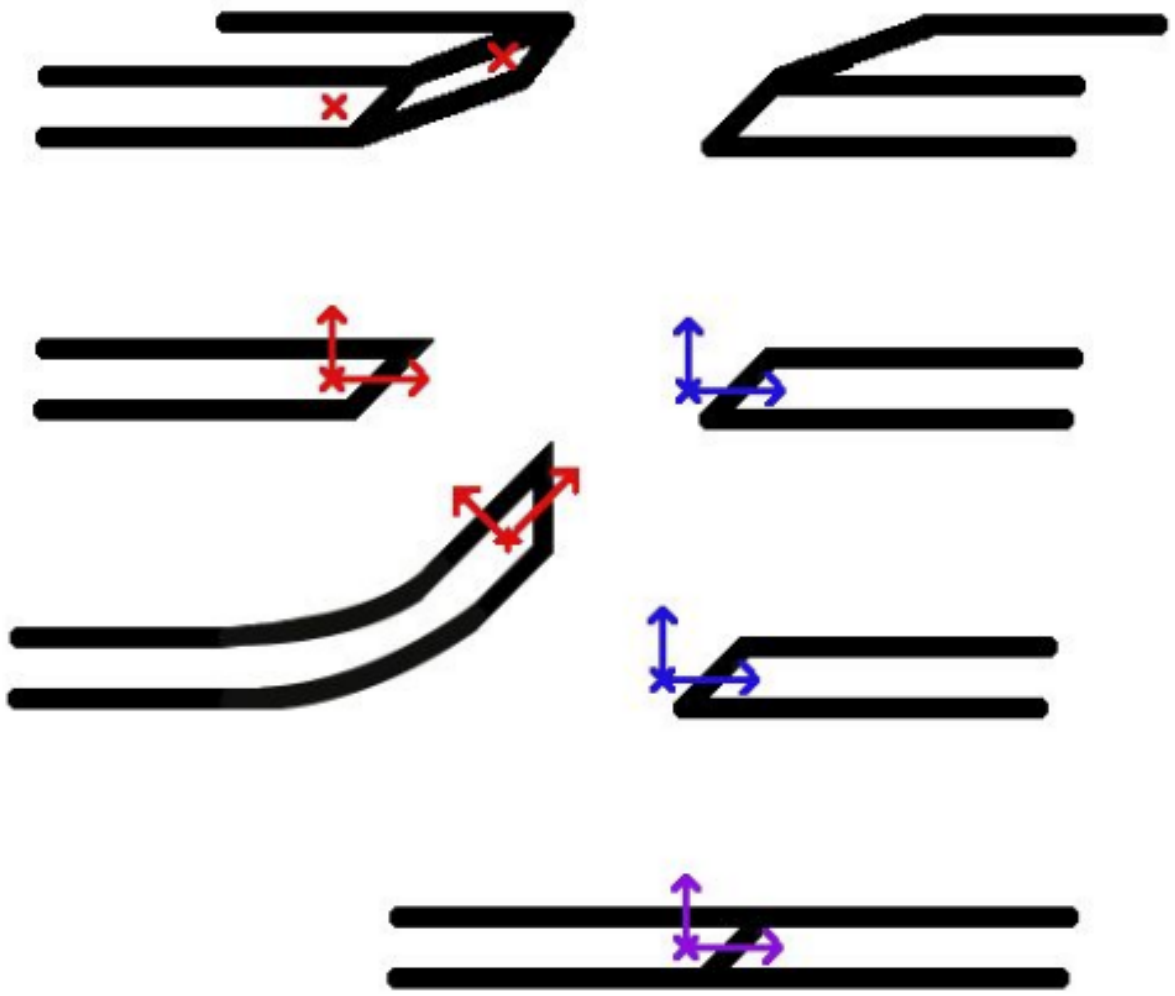

FIGURE 1.3: Représentation schématique de la tâche

— La gomme est noire, non texturée et brillante, ce qui implique des défis particuliers liés à la perception visuelle.

Tous ces défis font que la tâche n'a pas été automatisée lors du lancement du projet – d'autant plus avec des robots industriels, dont la délicatesse n'est pas la caractéristique première.

1.2 Présentation des bancs de test

Il est souvent nécessaire d'utiliser du matériel pour conduire des expériences de robotique. En particulier, dans le problème qui nous intéresse, nous devons tester des modules liés au cas d'usage venant à la fois de partenaires académiques et industriels. Ainsi, nous avons installé deux bancs de test. Le premier est le banc de test industriel, développé par l'équipe Michelin. Le second est la réplique que nous avons mise en place dans le laboratoire de l'Institut Pascal. Les choix matériels ont été imposés d'une part par le matériel disponible et d'autre part par les contraintes du projet.

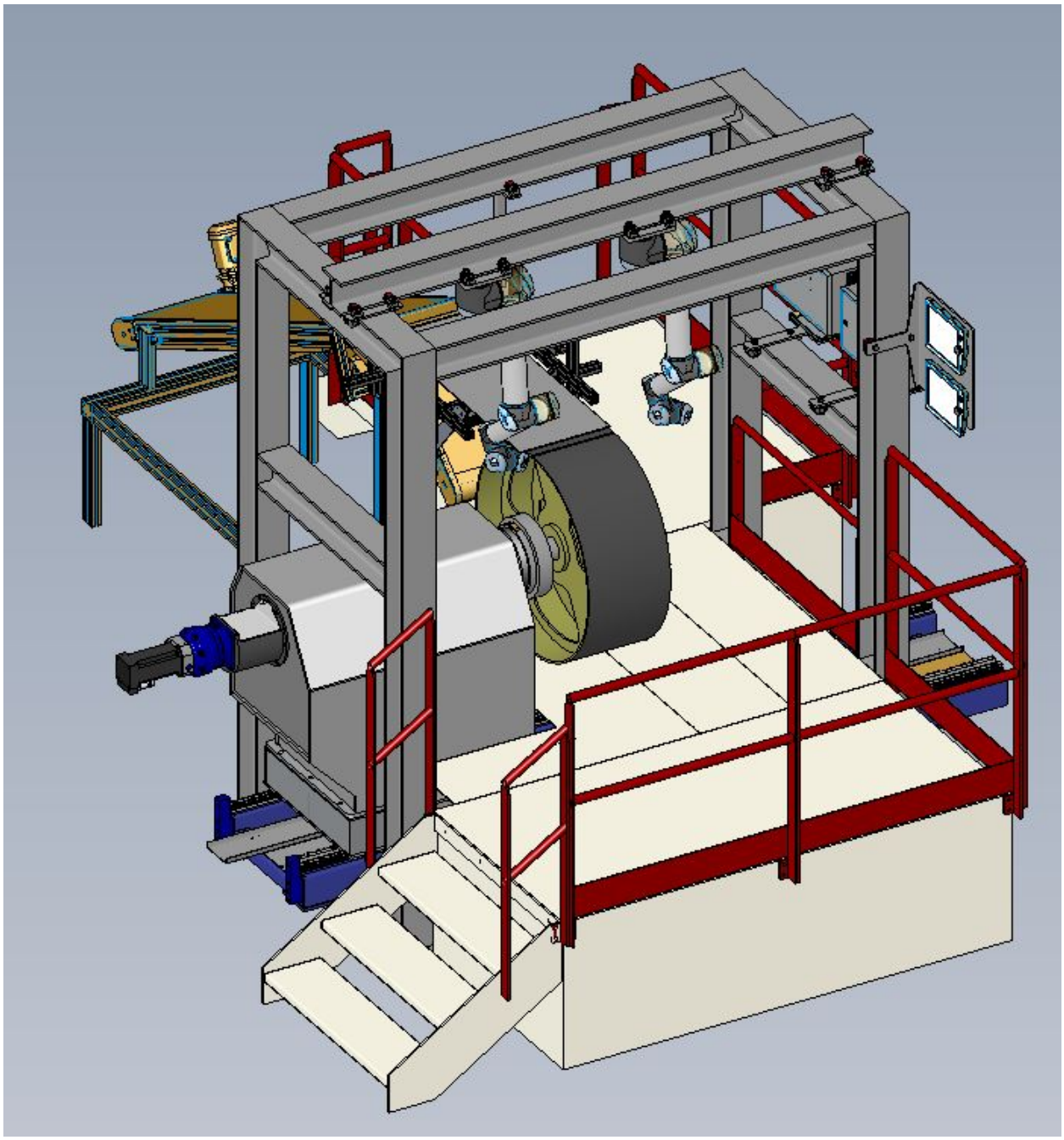

FIGURE 1.4: CAD du banc de test industriel

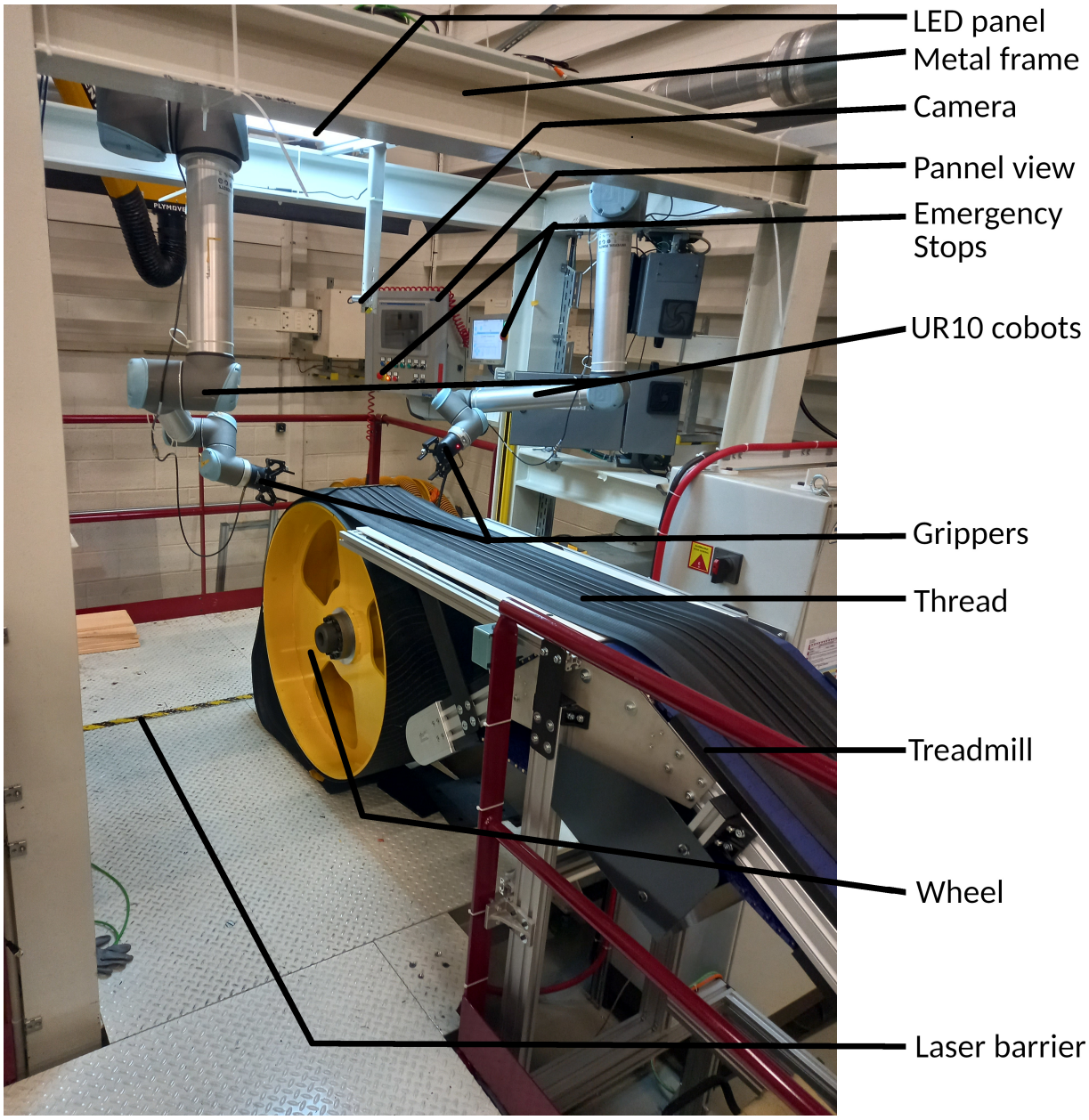

FIGURE 1.5: Illustration labellisée du banc de test industriel

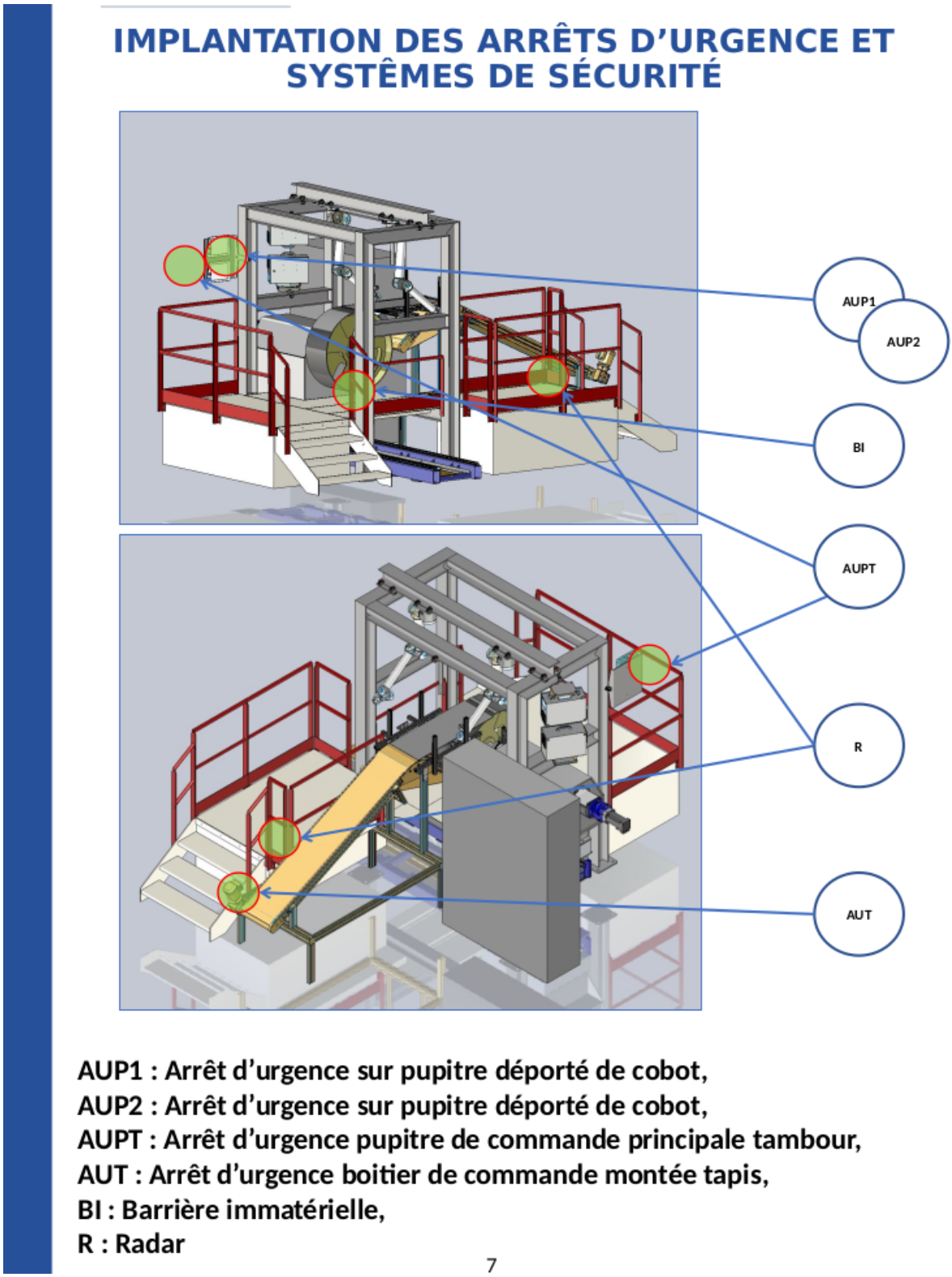

FIGURE 1.6: Synoptique de sécurité

Extrait du document Michelin de sécurité de la cellule.

1.2.1 Banc de test industriel

Fonctionnalités Le banc de test, modélisé Fig. 1.4 et montré Fig. 1.5, comporte deux bras robotiques remplaçant un opérateur humain sur une machine existante. Une partie de la conception

est donc imposée par la mécanique de la machine existante. Les préhenseurs sont fournis par le partenaire Zimmer. Au cours du projet, nous avons testé différents modèles, des séries électriques et pneumatiques. Sur la machine existante, le cycle est géré par un PLC (Programmable Logic Controller), ce qu'il faudra prendre en compte pour la solution finale. La roue sur laquelle la bande de roulement est posée – le pneu en construction dans le cas réel – est motorisée et pilotée par le PLC. Un tapis roulant alimente la machine en gomme. Pour résoudre la tâche, nous avons monté les deux bras UR10 à l'envers sur le cadre métallique. Pour améliorer l'efficacité des algorithmes de perception, nous avons ajouté une dalle LED afin de mieux maîtriser les conditions de luminosité.

Spécificités Comme le banc doit être adapté à une machine existante, cela contraint les adaptations faisables et celles impossibles. Par exemple, un actuateur poussant la gomme par dessous pour une saisie plus facile est impossible : même si cela aurait aidé le banc de test, le pneu ne peut pas être mécanisé pour des simples raisons de commodité. Le cylindre a le diamètre d'un vrai pneu, et le cadre de métal a été dessiné pour s'adapter aux machines existantes. Le PLC est un Rockwell ControlLogix, similaire à ceux utilisés en usine pour des machines comparables.

Sécurité La sécurité du banc de test est gérée par le PLC. Le synoptique sécurité est montré dans la Fig. 1.6. A l'avant et l'arrière de la machine, des barrières laser permettent la détection de l'opérateur dans la cellule robotique, empêchant les opérations de rotation de la roue et imposant le mode réduit aux cobots. Plusieurs arrêts d'urgence sont situés tout autour de la machine, pour le bien être pérenne à la fois des utilisateurs et des robots.

1.2.2 Banc de test académique

Spécificités Ce banc de test, créé dans le Hall Robotique de l'Institut Pascal, est montré dans la Fig. 1.7. Ce banc de test utilise aussi des robots UR10 ; les bras font cependant partie des robots manipulateurs sur base mobile Campero conçus par Robotnik. Dans notre cas d'usage, nous n'avons pas besoin d'exploiter la mobilité de la plateforme et avons par conséquent solidarisé les deux bases mobiles avec un profilé en aluminium, permettant d'installer une caméra dans une configuration similaire à celle du banc de test industriel. Les préhenseurs utilisés sont les grippers du concepteur Robotiq 2F (two fingers, deux doigts) et 3F (three fingers, trois doigts). Michelin nous a fourni une section de cylindre dont le diamètre est celui de la roue motorisée, ainsi que des morceaux de bande de roulement.

Différences entre les bancs de test Le banc d'essai académique présente quelques différences par rapport au banc d'essai industriel.

- Tout d'abord, la sécurité. Il n'y a pas de barrière laser ni de PLC, ce qui signifie qu'il n'y a pas d'arrêt d'urgence global. Ensuite, l'orientation des robots : sur le banc d'essai industriel, les robots sont montés tête en bas, avec les bases à 180° l'une par rapport à l'autre. Sur le banc d'essai académique, la configuration est identique car deux machines identiques sont stationnées en parallèle.
- Ensuite, les contraintes de l'espace de travail. Les structures sont différentes, ce qui signifie que l'encombrement l'est aussi.
- Enfin, le caoutchouc lui-même est différent. Au lieu d'une bande unique entourant le pneu, deux morceaux de gomme relativement petits permettent de simuler la tâche. Cette gomme a été utilisée pendant plusieurs années au lieu de quelques jours, ce qui entraîne une vulcanisation partielle, rendant le caoutchouc plus rigide et perdant ses propriétés élastiques.

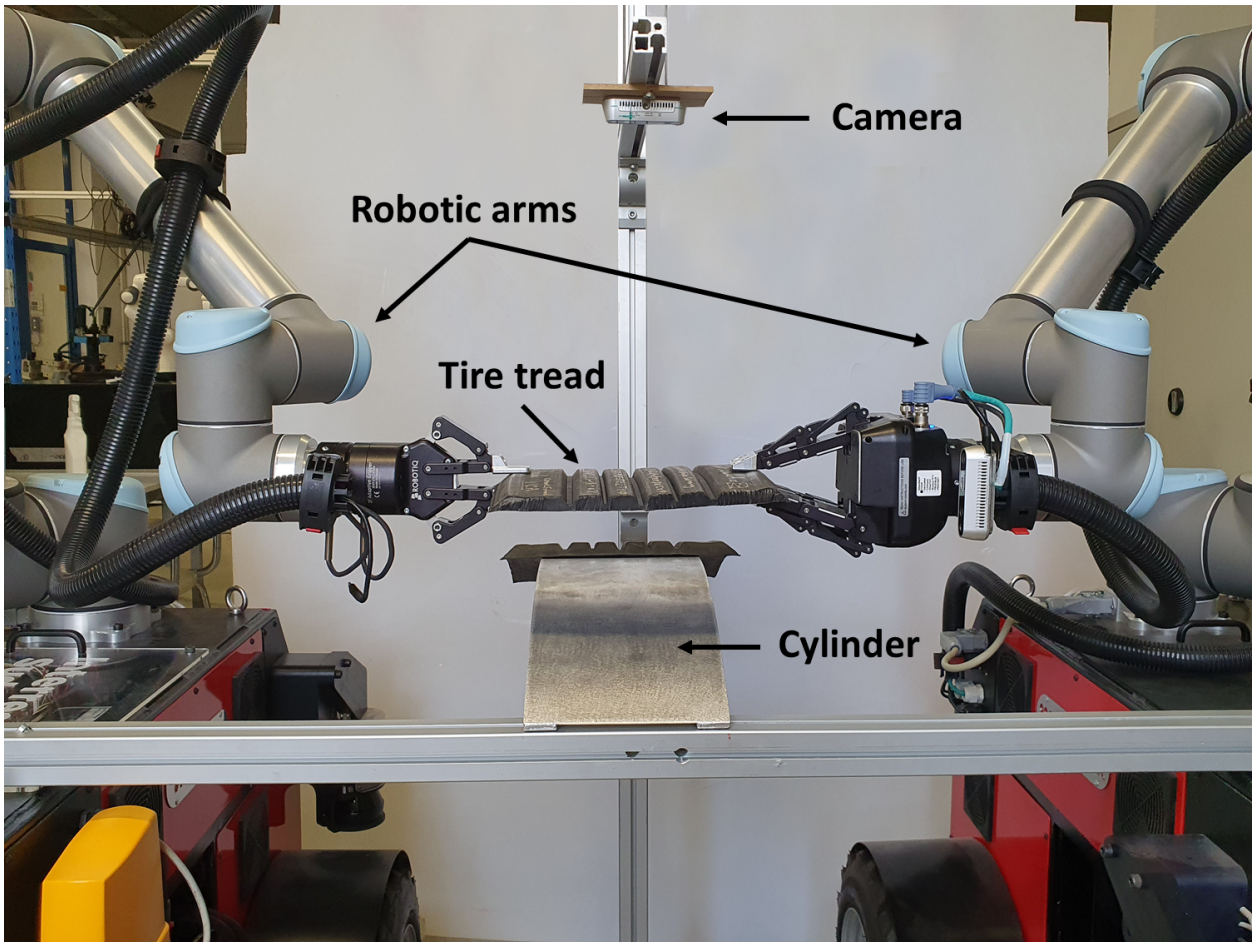

FIGURE 1.7: Banc de test académique

Autres bancs de test Dans certaines expériences, comme dans la section 3.4.3, nous utilisons d'autres installations. Cependant, elles sont toujours des adaptations mineures d'un des bancs de test présenté, adapté aux expériences que nous souhaitons conduire.

1.3 Méthodologie de développement logiciel

La méthodologie de développement logiciel utilisée est la méthodologie Agile [1], en accord avec le postulat épistémologique énoncé dans l'annexe A. Nous avons également appliqué le procédé nommée "Dogfooding" [Har06], signifiant dans ce contexte utiliser les plateformes développées pour les expériences et fréquentes démonstrations au laboratoire en utilisant la base de code du projet.

Nous suivrons les bonnes pratiques C++. Premièrement, nous appliqueront le paradigme Ressource Acquisition Is Initialization (RAII) [2] . RAII signifie que le constructeur est responsable de l'acquisition des ressources, comme l'allocation mémoire. Le destructeur est responsable de la libération des ressources. Habituellement, les objets sont alloués dans la pile (allocation statique), ce qui fait que la durée de vie de l'objet est liée au scope de sa déclaration. En cas de besoin d'allocation dans la pile (allocation dynamique), nous préférerons l'utilisation de smart pointers. Ces derniers fonctionnent en comptant les références à un même objet. Quand le smart pointer, ou l'objet contenant le smart pointer, sort du scope, le compteur est décrémenté. Quand le nombre

1. https://agilemanifesto.org/
2. https://www.stroustrup.com/bs_faq2.html#finally

de référence atteint 0, plus aucun objet n'a besoin du pointeur, ce qui est détecté ; cela déclenche l'appel du destructeur. Le paradigme RAII permet d'éviter les fuites mémoires et les pointeurs fous sans entraver les capacités bas-niveau du langage comme le font une machine virtuelle et un ramasse-miettes.

Nous suivrons également le paradigme SOLID [Mar00], l'acronyme anglais de :

— Single-responsibility principle : "There should never be more than one reason for a class to change."
— Open–closed principle : "be open for extension, but closed for modification."
— Liskov substitution principle : "Functions that use pointers or references to base classes must be able to use objects of derived classes without knowing it."
— Interface segregation principle : "Clients should not be forced to depend upon interfaces that they do not use."
— Dependency inversion principle : "Depend upon abstractions, [not] concretions."

Toujours soucieux de respecter les bonnes pratiques, nous utiliserons les patrons de conceptions [Gam+95] quand nécessaire, tout en évitant les "anti-pattern" [Sta10].

1.4 Conclusion

Dans ce chapitre, nous avons présenté un procédé industriel intervenant dans la fabrication de pneumatiques pour poids lourd. Ce procédé est le cas concret mettant en avant les difficultés à surmonter pour l'usage industriel. Ces difficultés regroupent :

1 - La modélisation de la déformation de la bande de gomme

2 - La perception de sa forme

3 - L'asservissement de forme à mettre en place

Une autre difficulté provient du choix du procédé à employer : en particulier, suite à la prolifération des technologies d'IA, ne vaut-il pas mieux utiliser un réseau de neurone pour répondre à la problématique ? Ou bien les stratégies spécifiquement développées pour le contexte de l'objet déformable restent des choix supérieurs ? Enfin, les dernières difficultés identifiées relèvent des moyens à déployer pour créer une solution permettant de répondre à ces problèmes :

4 - Quelles architectures de systèmes permettraient l'interfonctionnalité des techniques précédentes ?

Parmi ces difficultés, certaines sont de véritables verrous technologiques que nous nous efforcerons de résoudre. Leur identification précise nécessite une étude approfondie de l'état de l'art.

Chapitre 2

État de l'art

Notre problème consiste à réaliser une tâche d'assemblage impliquant un objet déformable. Dans la littérature, certains problèmes similaires ont été résolus. Le problème de manipulation d'un objet déformable saisi à l'aide de bras robotique peut être subdivisé en plusieurs étapes nécessaires : la modélisation de la déformation, la perception de la forme courante, et la loi de commande permettant l'asservissement de forme. Ces étapes ne sont bien sûr pas une vérité absolue applicable dans tous les cas ou toutes les techniques de manipulation d'objets déformables. En particulier, les réseaux de neurones peuvent aboutir à des solutions holistiques, combinant plusieurs de ces fonctions. Par exemple, la plupart des techniques basées sur les réseaux de neurones fusionneront la modélisation de la déformation et la loi de commande de la déformation dans leur implémentation. Pour des raisons de clarté, nous associerons le vocabulaire suivant :

— les techniques de **modélisation** seront appelées **approches**.
— les techniques de **perception** seront appelées **méthodes**.
— les techniques de **contrôle** seront appelées **stratégies**.

Notre problème se situe à l'intersection de plusieurs domaines : la vision par ordinateur, l'infographie, l'intelligence artificielle, les systèmes de contrôle et encore, bien sûr, la robotique, qui sont autant de sources de références valables pour la tâche. Comme le domaine est en pleine expansion, des travaux très pertinents ont été publiés après le début de nos recherches en 2020. Certains ont été développés, prototypés, testés ou portés en utilisant l'architecture proposée pour le projet SoftManBot. Nous les présenterons plus en détail dans le chapitre 4. Cette intersection de domaine crée donc la nécessité d'une interconnexion de techniques provenant de mondes avec des paradigmes complètement différents. La question de l'architecture logicielle à déployer se pose donc également.

Le reste de la bibliographie est organisée comme suit :

— La section 2.1 présente les différentes approches pour reconstruire le modèle de l'objet déformable.
— La section 2.2 présente les moyens de reconstruction de la forme d'un objet déformable perçu par des caméras.
— La section 2.3 présente les lois de commandes pertinentes vis-à-vis de la problématique de la thèse, ainsi que les stratégies existantes d'asservissement de forme d'objets déformables.
— La section 2.4 présente les architectures logicielles adaptées à la robotique et au cas d'application de l'objet déformable.

2.1 Modélisation de déformation

Les objets déformables peuvent être divisés en plusieurs catégories, comme énoncé dans [San+18]. Le premier concept à expliciter est la résistance à la compression. Un objet présente cette propriété s'il présente une résistance lorsque les extrémités opposées sont poussées l'une vers l'autre. Un objet déformable uniparamétrique sera appelé "linéique". Un objet déformable linéique avec une résistance à la compression sera catégorisé comme une poutre, un câble lorsqu'il ne présente pas de résistance à la compression. Un objet déformable biparamétrique sera appelé "planaire". En fonction de la présence ou non de résistance à la compression, l'objet sera respectivement qualifié de surfacique et de type tissu. Un objet déformable triparamétrique sera appelé volumique. Nous étudierons le type d'objet avec lequel nous travaillerons dans la section 2.1.6

Il y a plusieurs manières de représenter un objet déformable. L'étude systématique [NVP18] sépare les représentations en deux catégories : les approches basées sur un modèle et les approches sans modèle. Dans le cas de la présence d'un modèle, il peut être physique, géométrique ou encore appris.

Dans une application d'asservissement de forme, les modèles sont généralement formulés sous la forme d'une Jacobienne de déformation. Les matrices Jacobiennes ont une utilisation étendue dans les applications robotiques. Par exemple, dans le livre [Nak90], les auteurs présentent l'approche basée sur la Jacobienne pour le problème de la cinématique inverse. Dans ce cas, la Jacobienne est construite à l'aide de dérivées partielles, comme l'a écrit initialement [Whi69] et montrée dans l'équation 2.1 :

$$J = \begin{pmatrix} \frac{\partial x_1}{\partial q_1} & \frac{\partial x_1}{\partial q_2} & \cdots & \frac{\partial x_1}{\partial q_n} \\ \frac{\partial x_2}{\partial q_1} & \frac{\partial x_2}{\partial q_2} & \cdots & \frac{\partial x_2}{\partial q_n} \\ \cdots & \cdots & \cdots & \cdots \\ \frac{\partial x_m}{\partial q_1} & \frac{\partial x_m}{\partial q_2} & \cdots & \frac{\partial x_m}{\partial q_n} \end{pmatrix} \tag{2.1}$$

avec $J \in \mathbb{R}^{n \times m}$. $q_i \in \mathbb{R}$ est la position de l'articulation i, aussi appelée variable articulaire. $x_j \in \mathbb{R}$ représente la variable opérationnelle j, dépendante du système.

Cette matrice Jacobienne est une matrice de sensibilité, c'est à dire de la variation de la sortie i en fonction de la variation de l'entrée j. Dans le cas d'un objet déformable, la Jacobienne de déformation est la matrice qui relie le mouvement de l'effecteur aux variations de l'état de l'objet dans la représentation choisie. Chaque ligne correspondant à un Degré De Liberté (DDL) de l'effecteur et chaque colonne correspondant à un DDL de la représentation d'état de l'objet.

2.1.1 Modèles physiques

Les modèles physiques des objets déformables sont basés sur les lois de Newton [New33]. L'objet est représenté par un réseau de points appelé maillage. Les équations physiques décrivent l'interaction des points de l'objet les uns avec les autres.

2.1.1.1 Approche de modèle masse-ressort

Un modèle particulier de cette approche est appelé modèle masse-ressort (Mass-Spring Model). Chaque nœud, associé à une masse, est relié uniquement aux nœuds voisins par une force. Cette force est composée d'un ressort et d'un amortisseur. Le résultat de cette approche est illustré Fig. 2.1.

Dans cette représentation, la déformation de l'objet dépend de la configuration du réseau de nœuds. Les paramètres sont le nombre de nœuds, la position initiale et les connexions, définies

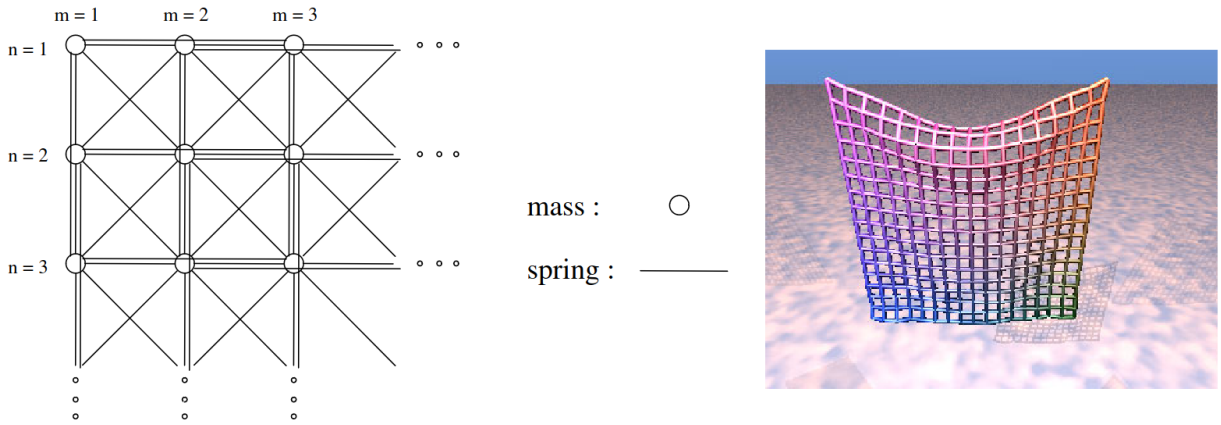

FIGURE 2.1: Illustration de modèle masse-ressort et son application

Extraits du papier [Pro+95]

par la rigidité du ressort et la puissance de l'amortisseur. Dans [Ter+87] et [TF88], des travaux pionniers dans le domaine du modèle masse-ressort présentent une approche dynamique et réaliste novatrice pour l'animation par ordinateur, en particulier pour la déformation élastique. L'année suivante, les éléments ressorts sont complétés par des unités de frottement visqueux et de glissement plastique, ce qui permet de modéliser des comportements viscoélastiques et plastiques.

Les principaux avantages de cette approche sont la rapidité de calcul et la facilité de compréhension. Ces points forts ont largement contribué à l'utilisation répandue de l'approche du modèle masse-ressort. Des exemples ont vu le jour, utilisant cette approche pour la représentation du comportement des tissus [Pro+95], la simulation des résultats de chirurgie plastique [SV+09] ou la modélisation des cheveux [SLF08].

Des développements relativement récents [Liu+13] ont permis d'obtenir une simulation approximative mais suffisamment proche en environ 50 millisecondes, ouvrant la voie à des systèmes en boucle fermée utilisant ces modèles.

L'un des inconvénients des modèles masse-ressort réside dans la difficulté de régler les paramètres. Dans [LSH07], ces paramètres sont identifiés pour chaque élément. L'identification est basée sur la matrice de rigidité obtenue à partir du modèle d'éléments finis de la structure. Dans [Bia+04], un algorithme génétique est utilisé pour générer le réglage du modèle.

D'autres inconvénients de ces approches comprennent une faible stabilité et un manque de conservation du volume. Ce dernier peut être atténué en incluant des ressorts virtuels [BC00], mais la stabilité demeure un problème d'intérêt.

2.1.1.2 Approche Dual-Quaternion

Une approche prometteuse pour la modélisation des objets déformables est la modélisation par quaternions duaux (DQ - Dual Quaternion). La modélisation par DQ utilise un ressort quaternion pour suivre l'orientation des nœuds. Avec les DQ, il existe une force de rappel associée au changement d'orientation [Aba+17].

Cette approche appliquée à la modélisation des objets déformables est limitée dans la littérature disponible. Une contribution notable est l'utilisation de cet outil mathématique dans [WY10] : l'approche des quaternions duaux est appliquée à l'origami, des panneaux rigides articulés autour de plis nommés sommets. En infographie, les utilisations les plus proches des quaternions duaux pour le problème d'intérêt sont utilisées dans le "skinning" d'un modèle déformable par un squelette [Kav+07], ou pour la fusion de transformations rigides [Kav+06a].

2.1.1.3 Approche éléments finis

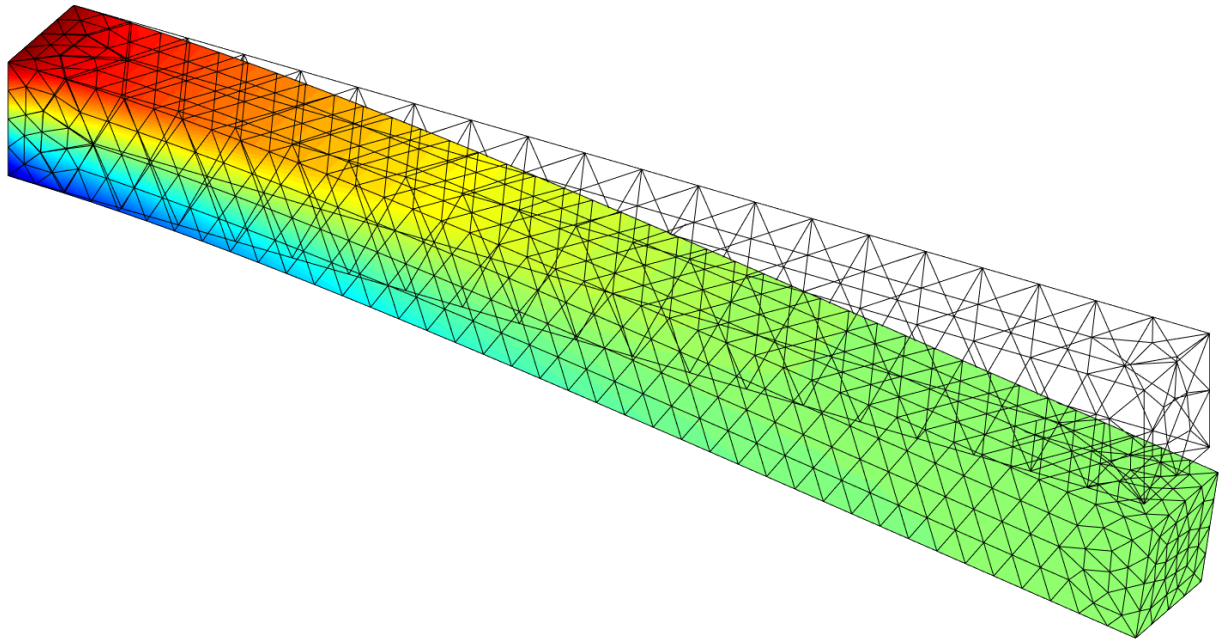

FIGURE 2.2: Illustration de modèle élément finis

Une autre approche est la méthode des éléments finis (FEM - Finite Element Method). Cette approche est globale et peut être utilisée pour calculer d'autres éléments que la forme d'objets souples, tels que la distribution de chaleur.

Dans cette approche, l'objet est découpé en un maillage de nœuds. Les nœuds sont reliés les uns aux autres pour former des éléments, comme illustré Fig. 2.2. Souvent, les éléments sont des triangles, mais dans certains cas, des tiges, des tétraèdres et d'autres géométries simples sont utilisées. Le problème est ensuite formulé à l'échelle de l'élément en interpolant les champs recherchés en fonction de leurs valeurs aux points nodaux, en incluant les conditions aux limites. Les éléments sont ensuite fusionnés pour produire le résultat final [LB13].

Les principaux atouts de la FEM sont sa polyvalence et sa précision, tout en restant facilement calculable, principalement grâce à la parallélisation massive rendue possible par ces modèles [Arg+18] [Pan05]. La FEM a été largement utilisée dans les approches de perception et de manipulation d'objets déformables. En particulier, l'approche FEM est utile pour suivre l'état des objets déformables à l'aide de caméras [WLS12] [AMN17] [SKM19a].

Dans [Due+18], la FEM est utilisée pour modéliser plusieurs objets en mousse. Ces objets, saisis par un robot bi-bras, sont ensuite manipulés pour atteindre une forme cible. Cependant, le système n'est pas en boucle fermée et le calcul de la configuration articulaire cible prend environ 2 secondes. La boucle ouverte a pour conséquence des instabilités de commande : l'objet prend des "formes alternatives" pour une configuration de préhension correspondant à la forme cible. Ces formes alternatives résultent de la non-unicité de la pose énergétiquement minimale de l'objet sous contrainte, considérée comme la plus stable.

L'inconvénient majeur de cette approche est le lien entre la complexité algorithmique et le nombre d'éléments, ce qui peut être aisément testé avec des sources disponibles en ligne [1]. La

1. Peter van Alem. Introduction to FEM, `https://www.mathworks.com/matlabcentral/fileexchange/46384-introduction-to-fem`, MATLAB Central File Exchange.

puissance de calcul nécessaire liée au nombre d'éléments – l'ordre de complexité exact dépend de la composition exacte des éléments, avec un ordre $O(n^4)$ pour des tétraèdres, avec un grand nombre d'éléments nécessaires pour modéliser des objets complexes présentant des irrégularités.

2.1.1.4 Vue d'ensemble des modèles physiques

Le tableau 2.1 donne quelques éléments de comparaison des modèles physiques de déformation.

TABLE 2.1: Table de comparaison des modèles physiques

Approche	**Avantages**	**Inconvénients**
Masse-Ressort	Conceptuellement simple Performant pour les grandes déformations	Potentiellement instable Nécessite du paramétrage
Dual Quaternions	Léger en calcul	Moins générique. Conceptuellement difficile
Éléments Finis	Grande précision	Lourd en calcul

2.1.2 Modèles géométriques

Les modélisations géométriques des objets déformables reposent sur une correspondance géométrique de forme de l'objet. La forme sera soumise à des transformations mathématiques. Les approches de cette famille sont plus légères en termes de calcul, mais doivent être choisies et réglées avec soin pour correspondre à la réalité, car elles ne sont pas basées sur des principes physiques. Le comportement mécanique du système est implicite dans ces modèles géométriques.

2.1.2.1 Position Based Dynamics

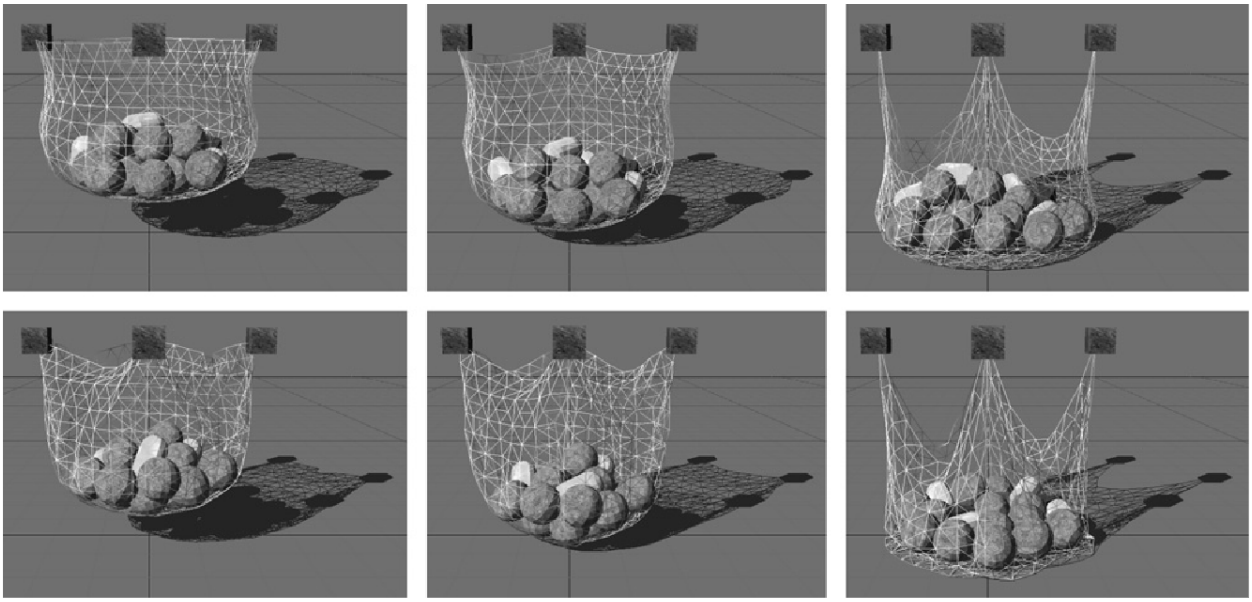

FIGURE 2.3: Illustrations générées avec Position Based Dynamics

Extrait du papier [Mül+07]

Dans la section précédente, des approches bien établies telles que la FEM ou le modèle de masse-ressort calculent la déformation de l'objet en fonction des vitesses internes des points. Mais que se passerait-il si la simulation de déformation était directement basée sur la position ? Dans [Mül+07], les auteurs ont présenté une telle approche, appelée Position Based Dynamics (PBD). L'approche PBD travaille directement sur les positions des nœuds, améliorant considérablement la contrôlabilité et permettant de gérer les collisions et les pénétrations d'objets. L'approche consiste à définir un ensemble de contraintes géométriques - des fonctions qui doivent donner des inégalités ou des égalités sur des nœuds donnés - sur l'objet. Les contraintes sont pondérées par un facteur de rigidité, ajustant leur effet sur le système global. Ensuite, à chaque étape, l'état de l'objet est calculé de manière itérative.

Les avantages de l'approche PBD sont sa rapidité, sa stabilité et sa contrôlabilité. En ajustant l'ensemble de contraintes, des comportements tels que la collision, la compression ou le déchirement des vêtements sont visuellement plausibles, comme montré Fig. 2.3. Les limites de l'approche PBD sont celles de toute approche géométrique : la plausibilité ne signifie pas une précision physique. Ainsi, l'approche PBD est largement utilisée en infographie, dans les jeux vidéo et l'animation.

La méthode a été améliorée au fil du temps pour converger plus rapidement [Mül08], inclure une énergie dissipative [MMC16], ou créer un simulateur de chirurgie en réalité virtuelle [Pan+15].

2.1.2.2 Meshless Shape Matching

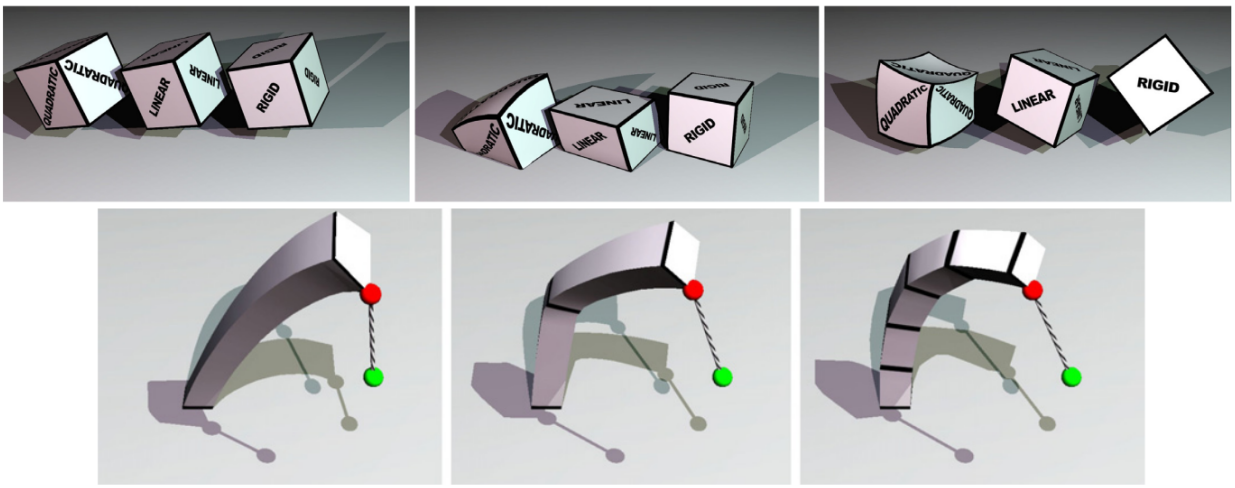

FIGURE 2.4: Illustrations générées avec Meshless Shape Matching

Extrait du papier [Mül+05]

Bien que la PBD soit plus rapide à calculer que la FEM, elle itère plusieurs fois sur tous les nœuds. Dans [Mül+05], les auteurs présentent l'approche Meshless Shape Matching (MSM). Cette approche divise l'objet en grappes de points. Les déformations sont appliquées de manière globale, géométrique, sur ces grappes. Plusieurs transformations peuvent être linéaires, simulant des objets quasi-rigides, ou quadratiques, rendant l'objet plus déformable. La modularité introduite par les grappes permet des déformations visuellement réalistes telles que la flexion de barres ou les déformations plastiques. Ceci est illustré dans la Fig. 2.4.

Certaines utilisations pertinentes de cette approche incluent la création de retour haptique [Tia+13] ou la modélisation uniforme de fluides ainsi que d'objets volumiques déformables [Mac+14]. Dans [Gul+15], les auteurs utilisent le MSM pour estimer en ligne la déformation d'un objet, ce qui est amélioré en ajoutant des données réelles issues d'une FEM [Gül+17].

Cette approche se distingue parmi les modèles géométriques : elle nécessite une puissance de calcul encore plus faible, est efficace en terme de mémoire utilisée et est stable de manière inconditionnelle. Cela permet la simulation de centaines d'objets déformables en temps réel, y compris avec gestion de collisions. De plus, dans le problème d'intérêt de cette thèse, le MSM peut produire les positions des points cibles pour atteindre la forme souhaitée et calculer les points optimaux de saisie de l'objet dans sa configuration initiale.

2.1.2.3 As-Rigid-As-Possible

As-Rigid-As-Possible (ARAP) [SA07] est une approche d'infographie pour les objets déformables avec une particularité : l'objet ne souhaite pas être déformé. Dans cette approche, la surface de l'objet est divisée en petites cellules qui se chevauchent. Les cellules sont soumises à la fois à des contraintes externes et à une énergie élastique représentant la rigidité. Le problème devient alors un problème d'optimisation non linéaire. La déformation est calculée de manière itérative, convergeant en moins de cinq itérations. Cela donne une approche simple, robuste et efficace.

Cette approche a été utilisée pour améliorer les algorithmes de perception [Han+18], y compris la reconstruction de forme à partir d'un modèle [Par+15], et réécrite sous forme basée sur les arêtes [II09].

Une approche proche est As-Similar-As-Possible [LG12] [Jia+17]. Cette approche repose sur une formulation similaire pour détecter les caractéristiques d'une image basée sur un modèle original. Ensuite, l'approche les ajuste le plus précisément possible au modèle, d'où le terme "Similar" dans le nom.

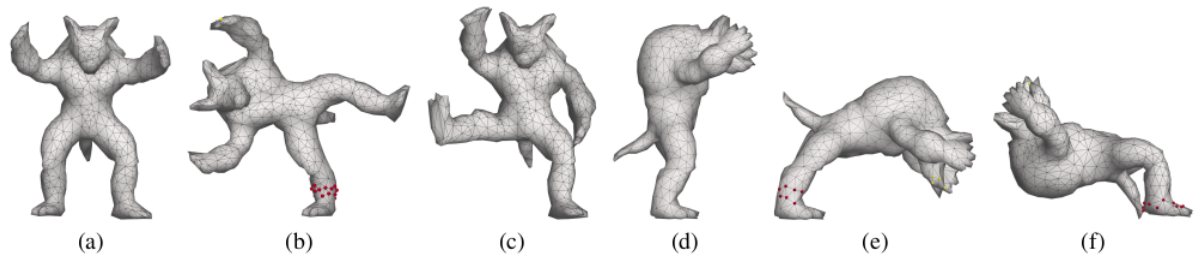

FIGURE 2.5: Objet complexe dont les déformations sont simulées avec ARAP

Extrait du papier [SA07]

2.1.2.4 Vue d'ensemble des modèles géométriques

Le tableau 2.2 donne quelques éléments de comparaison des modèles géométriques de déformation.

TABLE 2.2: Table de comparaison des modèles géométriques

Approche	**Avantages**	**Inconvénients**
Position-Based Dynamics	Très polyvalente	Réglages plus complexes
Meshless Shape Matching	Très rapide grâce à la clusterisation	Approche la moins précise
As-Rigid-As-Possible	Grande précision en tant que approche pseudo-physique	Relativement plus coûteuse

2.1.3 Approches par apprentissage

La modélisation des objets déformables à l'aide de réseaux de neurones. consiste à utiliser un réseau de neurones pour prédire la forme de l'objet. Par exemple, dans l'article [HWK19], un réseau de neurones est utilisé pour obtenir un modèle non linéaire d'actionneurs souples, ce qui évite le calcul complexe des variables d'état. De même, dans l'article [Yan+20], un réseau neuronal est utilisé pour estimer l'état d'une corde, en mettant à jour les variables d'état en temps réel, puis en manipulant la corde pour lui donner une forme souhaitée.

Cette approche est relativement rare dans le contexte de la manipulation d'objets déformables, où les réseaux de neurones sont plus souvent utilisés pour la perception, le contrôle ou les deux, plutôt que pour le calcul du modèle. Cependant, cette approche est intéressante pour résoudre le problème de l'estimation non linéaire de l'état. Un des problèmes de cette approche réside dans l'apprentissage, qui n'est que peu généralisable et loin d'être trivial à mettre en œuvre. Le principal inconvénient réside dans la donnée : il n'existe pas de base de donnée de déformation d'objet, sachant que chaque objet nécessiterait une base dédiée. De plus, il existe la question de la boîte blanche [ABPZ21] concernant la compréhension du fonctionnement interne de cette approche.

2.1.4 Approches sans modèle

Les approches sans modèle sont des approches qui ne nécessitent pas de modélisation ou de simulation des déformations. Ces approches reposent généralement sur l'estimation en ligne de la jacobienne de déformation, en utilisant des techniques pour maintenir l'intégrité structurelle de l'objet [Ber13]. Par exemple, dans [Ala+18], la jacobienne de déformation de l'objet est reconstruite autour des zones de préhension en utilisant des points caractéristiques pour résoudre un problème d'optimisation. Ces approches sans modèle permettent de traiter des objets présentant des propriétés physiques non linéaires et variables dans le temps.

Une autre façon de réaliser une approche sans modèle est d'utiliser un réseau de neurones. Par exemple, dans [Che+20], les robots apprennent à modeler du sable à partir de démonstrations humaines sans utiliser de modèle, tandis que dans [Wu+20a], les auteurs se concentrent sur l'approche sans modèle en utilisant l'apprentissage par renforcement et en évitant explicitement l'apprentissage à partir de démonstrations.

2.1.5 Vue d'ensemble des modèles

Le tableau 2.3 synthétise les différences principales entre les approches de modèle présentées dans cet état de l'art.

TABLE 2.3: Table de comparaison des modèles

Approche	Avantages	Inconvénients
Basé physique	Physiquement correct	Précision exponentiellement liée au coût calculatoire
Basé géométrique	Léger en calcul Contrôlable et stable Versatile	Nécessite du paramétrage
Basé apprentissage	Gère facilement la non-linéarité	Nécessite entraînement Nécessite dimensionnement Résultat en boîte blanche
Sans modèle	S'adapte à l'indicateur pertinent pour la tâche	Difficile à généraliser

2.1.6 Réduction de l'ordre du problème

Dans notre cas d'utilisation, la bande de roulement est épaisse, avec une résistance à la compression. Ainsi, nous nous situons dans la catégorie des "volumiques". Notre matériau peut être considéré comme isotrope, avec des propriétés élastiques et plastiques. La tâche nécessite d'éviter les déformations plastiques. Il existe des effets mécaniques dus à la géométrie de l'objet, principalement les sillons, créant une anisotropie globale. A priori, il ne semble pas y avoir de réduction d'ordre triviale possible. Cependant, si l'épaisseur peut être négligé, tout comme les différences de flexion induites par la géométrie, notre problème peut être réduit à un problème isotropique d'asservissement de forme surfacique. C'est effectivement le cas, comme cela a été validé expérimentalement dans la section 4.4.4.

2.2 Méthodes de perception informatique

Le contrôle et la modélisation sont inutiles sans capteurs. Dans notre cas, nous avons besoin d'une méthode de détection de forme en ligne pour estimer la forme d'un objet. Cette méthode peut être analytique, c'est à dire basée sur des équations ou des algorithmes, ou basée sur un réseau de neurones, c'est à dire fonctionnement avec un entraînement selon un principe de boîte blanche - un résultat fonctionnant selon une méthode dont on connaît tous les rouages, mais sans en connaître l'explication globale et les limitations inhérentes.

2.2.1 Méthodes de perception analytique

Dans cette section, nous nous concentrerons sur des méthodes non basées sur des réseaux de neurones pour l'estimation de forme en ligne à l'aide de capteurs visuels.

Avec une caméra monoculaire, des travaux importants ont été réalisés en utilisant la méthode de "Shape-from-Template" (SfT) [Bar+15]. Cette méthode permet de reconstruire la forme 3D de l'objet en utilisant un modèle qui contient la forme et la texture de repos de l'objet, illustré dans la Fig. 2.6. La méthode a été étendue pour améliorer la vitesse [ÖB17] ou avec un modèle intégré [Par+15].

Les méthodes générales de suivi d'objets déformables à l'aide d'une caméra RGB-D impliquent généralement une simulation de la déformation pour éliminer les valeurs aberrantes et

estimer la forme en prenant en compte l'auto-occultation. Dans [PLS15], les auteurs utilisent un nuage de points segmenté provenant d'une caméra RGB-D. Le nuage de points est ajusté à un maillage en utilisant la FEM pour modéliser l'élasticité et accomplir la tâche. Dans [TT22], les auteurs introduisent une régularisation dans leur méthode de suivi, qui respecte à la fois la structure locale et la topologie globale.

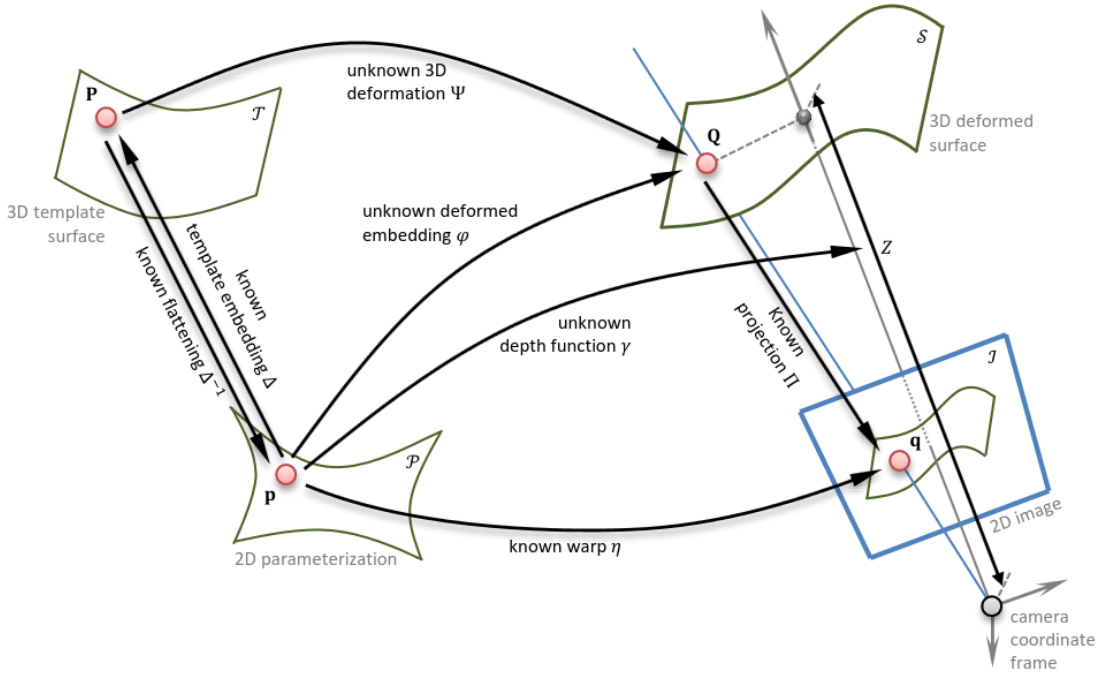

FIGURE 2.6: Illustration du fonctionnement de l'algorithme Shape-from-Template

Extrait du papier [Bar+15]

Certaines méthodes sont plus adaptées à des cas d'utilisation spécifiques. Par exemple, si l'objet peut être assimilé à une corde ou à une ceinture soumise à l'influence de son propre poids, il suit l'équation de "caténaire", issue du latin *catenaria*, en référence à une chaîne (*catena*). Il s'agit d'un problème bien connu, supposément publié pour la première fois par Gottfried Leibniz, Christiaan Huygens et Johann Bernoulli dans Acta Eruditorum en juin 1691 [Cli60]. Les solutions de l'équation de caténaire impliquent des cosinus hyperboliques, la différence de hauteur et les distances des points d'ancrage ainsi que la tension du matériau. Cette propriété a été utilisée pour suivre un morceau de caoutchouc et extraire des caractéristiques de forme, permettant de déterminer la tension interne de l'objet [Fil+22].

Le tableau 2.2 donne quelques éléments de comparaison des méthodes analytique de perception.

TABLE 2.4: Tableau comparatif des perceptions analytiques

Méthode	Capteur	Avantages	Inconvénients
Méthodes basées sur l'équation du Caténaire	Caméra RGB-D	Simple Rapide Précis	L'objet doit ressembler à un câble
Suivi 3D d'objet déformable	Caméra RGB-D	Méthode générique	Nécessite une segmentation du nuage de points
SfT	Caméra monoculaire	Rapide Précis	Nécessite un modèle de référence

2.2.2 Méthodes de perception par réseaux de neurones

Les réseaux de neurones [MP43] ont été conceptualisés il y a 80 ans. L'avènement du calcul parallèle grâce aux GPU a transformé ce concept en un véritable changement de paradigme à échelle individuelle. Un réseau neuronal est composé de neurones individuels. Le neurone peut être décrit comme une fonction mathématique $\mathbb{R}^n \rightarrow \mathbb{R}$. Les neurones sont organisés en couches, qui sont interconnectées pour former le réseau [GBC16] [LC19]. L'introduction de cette technologie a été une véritable révolution, en particulier pour la perception informatique.

2.2.2.1 Architectures des réseaux de neurones

L'architecture représente l'interconnexion précise des couches du réseau. Étant donné qu'un seul neurone peut recevoir un nombre colossal d'entrées, différentes approches ont été essayées au fil des ans pour la configuration des réseaux. Les réseaux modernes utilisent une architecture composée de couches denses, couches convolutives, et autres pour répondre au problème à résoudre.

Dense Neural Network Dans un Dense Neural Network (DNN), chaque neurone d'une couche est connecté à chaque neurone de la couche suivante. Dans le cas de la classification d'images, l'entrée est aplatie et donnée en tant qu'entrée à la première couche du réseau. L'aplatissement consiste en une vectorisation, transformant le tableau bidimensionnel de pixels en un vecteur unidimensionnel contenant les données. Les performances d'un réseau peuvent être améliorées en ajoutant plus de couches, ce qui a été à l'origine du principal problème de l'architecture, à savoir le vanishing gradient [Hoc98].

Convolutional Neural Network Dans un réseau Convolutional Neural Network (CNN), une variété de filtres est appliquée à l'entrée. Ces filtres utilisent une opération mathématique appelée convolution. Les successions de convolutions sont utilisées pour extraire des caractéristiques de l'image de base. Les réseaux convolutifs nécessitent un ordre de grandeur de moins de paramètres que les réseaux denses [Gu+18]. L'un des réseaux convolutifs les plus connus et largement utilisés est U-net [RFB15], illustré dans la Fig. 2.7. Ce réseau entièrement convolutif segmente rapidement les images pour des applications de servo-assistance et nécessite moins de labellisation lors des phases d'entraînement pour une précision accrue par rapport aux méthodes précédentes. Une autre architecture bien connue est Mask R-CNN [He+17]. Ce réseau segmente pixel par pixel une image, permettant la classification et la création de boîtes englobantes des caractéristiques présentes dans l'image.

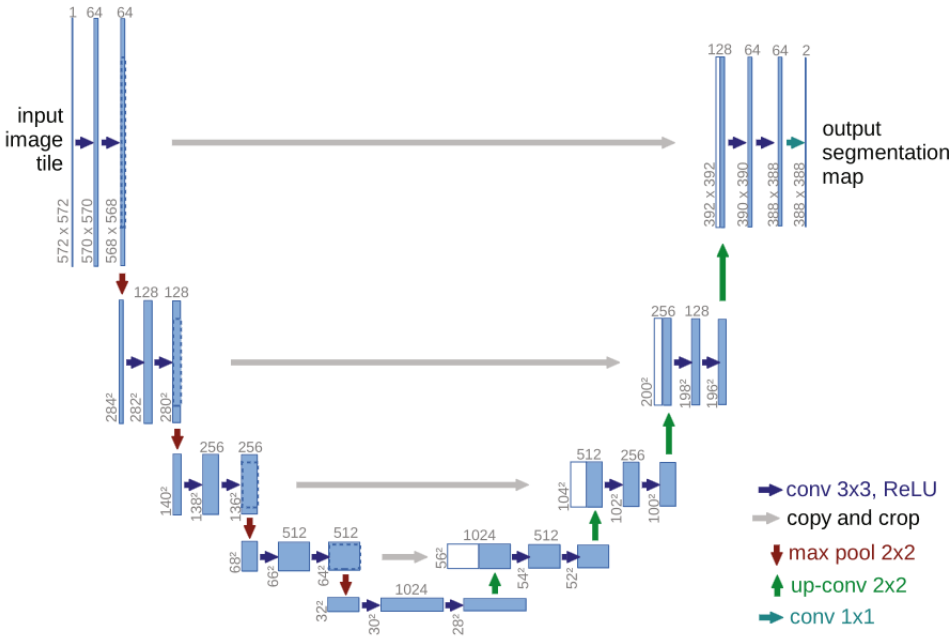

FIGURE 2.7: Architecture du réseau UNet

Extrait du papier [RFB15]. La forme de l'architecture en U est à l'origine du nom du réseau.

Graph Neural Network : Dans un Graph Neural Network (GNN), les données sont agrégées sous forme de graphe – ce qui implique que les Graph Neural Network sont des généralisations des Convolutional Neural Network. Ces réseaux propagent les informations le long des arêtes du graphe, permettant d'enrichir le contexte de chaque nœud [Wu+20b]. Cette architecture permet de travailler avec des données non structurées, comme par exemple les interactions des personnes sur un réseau social, la prédiction du trafic routier, la modélisation de systèmes physiques, ou le contrôle de robots [Zho+20].

Synthèse des architectures de réseaux de neurones existantes

Le tableau 2.5 donne quelques éléments de comparaison des principales architectures de réseaux de neurones.

TABLE 2.5: Tableau récapitulatif des architectures de réseaux de neurones existantes

Architecture	Avantages	Inconvénients
DNN	Générique. Garantie mathématiques de convergence	Complexité algorithmique
CNN	Particulièrement adaptée aux images	Nécessite des données structurées
GNN	Particulièrement adaptée aux données non structurés	Difficile à transférer

2.2.2.2 Techniques d'entraînement de Réseaux de Neurones

Les techniques principales d'entraînement de réseaux de neurones dans le cadre des travaux de robotique sont l'apprentissage supervisé et l'apprentissage par renforcement.

Dans le cas de l'apprentissage supervisé, un réseau de neurones est entraîné avec des données connues, et effectue une prédiction. Cette prédiction est comparée avec les résultats attendus (dis "labellisés"). L'erreur de sortie sera propagée de couche en couche par un algorithme appelé "rétropropagation". L'algorithme permet d'obtenir un gradient, métrique permettant de savoir comment modifier les poids et les biais des neurones afin d'obtenir le résultat attendu en sortie du réseau de neurones. En répétant cette opération sur toutes les données d'entraînement, il est possible d'appliquer une technique de mathématiques numériques, la descente de gradient. Les poids et biais sont adaptés : notre réseau aura appris [CC08].

Cette méthode souffre d'un problème majeur : le "vanishing gradient". Sur un réseau dense possédant un nombre important de couches, l'influence du gradient sera complètement diluée après la traversée et la propagation d'un nombre significatif de couches, empêchant l'adaptation des poids des premières couches du réseau. Le problème du "vanishing gradient" [Hoc98] a été résolu grâce à l'utilisation de "skip connections" en ajoutant l'entrée à la sortie d'une couche, ce qui permet à la rétropropagation du gradient de contourner plusieurs couches. Cette nouvelle technique a été introduite dans l'architecture des Réseaux Résiduels (ResNets) [He+16].

Dans le cas de l'apprentissage par renforcement, la démarche est inverse : le réseau interagit avec l'environnement, et on récompense les actions au lieu de pénaliser les erreurs [KLM96]. L'apprentissage peut se faire par des procédés aléatoires, ou bien avec un principe de gradient dans le cas de l'algorithme du gradient temporel [Mae11].

Parmi les procédés aléatoires, portons notre attention sur les algorithmes génétiques [For96]. Le principe de cette classe d'algorithmes est de générer une population avec des caractères aléatoires et d'attribuer à cette population une récompense basée sur l'adéquation avec un critère. On sélectionne une partie de la population répondant le mieux au critère, et on la fait se "reproduire" : en gardant un set de paramètres initiaux similaire, on introduit des mutations, changement aléatoire des dits paramètres. Ce type d'algorithme a été utilisé pour régler des paramètres de modèles physique [Bia+04], paramètrer des fonctions de façonnage de commande H_∞ [Yan+18], et bien sûr l'entraînement de réseaux de neurones [LS12] [MTK96].

Synthèse des techniques d'apprentissage en Machine Learning

Le tableau 2.6 donne quelques éléments de comparaison des techniques d'apprentissage en Machine Learning.

TABLE 2.6: Tableau récapitulatif des techniques d'apprentissage existantes

Technique	Avantages	Inconvénients
Supervisée	Nécessite uniquement de la donnée	Nécessite beaucoup de données Nécessite de la labellisation
Renforcement	Optimisation d'une fonction de récompense au lieu de minimiser une erreur de prédiction	Nécessite une fonction de récompense Nécessite un environnement Convergence longue
Algorithme génétique	Facile à mettre en place Pas besoin de gradient	Sensible aux conditions initiales Mauvaise scalabilité

2.2.2.3 Perception d'objets déformables avec des réseaux de neurones

Les réseaux de neurones ont été appliqués au cas d'usage de la perception d'objets déformables par vision informatique. En particulier, une approche de Shape-from-Template, Deep Shape-from-

Template, utilise un réseau profond intégrant les données du patron dans le poids des nœuds du réseau [FJ+22]. Les réseaux convolutifs ont également été appliqués pour la segmentation de multiples câbles dans un même environnement [Cap+22]. Il est également possible de percevoir l'objet déformable avec des moyens tactiles : les Convolutional Neural Networks sont particulièrement adaptés au traitement et à l'adaptation des données provenant de moyens haptiques [Gan+19].

2.2.2.4 Contrôle de robots par réseaux de neurones

Les réseaux de neurones ont été utilisés pour contrôler des robots depuis 35 ans [JC88]. Cependant, cette catégorie de stratégies de contrôle souffre d'un problème majeur : la reproductivité sur un système différent de celui utilisé pour la mise au point. Des efforts récents de recherche sont focalisés sur le Transfer Learning [TS09], en particulier son application aux tâches robotiques [Bea+21]. Cette méthodologie permet d'entraîner un réseau travaillant sur une tâche proche en réduisant le temps d'entraînement et la quantité de données nécessaires.

Dans le cadre de la manipulation d'objets déformables, nous pouvons citer les travaux cherchant à plier au mieux du linge [Avi+22] [Hie+22], manipuler des cordes [Den+22] [YSS22], ou encore des objets déformables linéique [Zak+22]. Dans [Avi+22], les auteurs ont utilisé une variante d'U-Net pour apprendre des primitives de pliage de linge, atteignant un taux de vêtement pliés de 40 par heure. Dans [Hie+22], les auteurs ont combiné le Reinforcement Learning à des stratégies de Learning from demonstration pour entraîner leur réseau. Dans [Den+22] et [Zak+22], les auteurs utilisent également le Reinforcement Learning. Enfin, dans [YSS22], les auteurs utilisent le Reinforcement Learning pour estimer en ligne le modèle dynamique de l'objet déformable manipulé afin d'appliquer une stratégie de contrôle MPC à l'objet.

2.3 Lois de commande pour l'asservissement de forme

Cette section présente les stratégies de contrôle d'une part, résultats connus parfois appliqués au cas de l'objet déformable, et les stratégies propres à l'asservissement de forme d'autre part.

2.3.1 Contrôle-commande classique

Le contrôle-commande est une science basée sur la rétroaction. De tels systèmes existent depuis plus de 2000 ans, comme les horloges à eau de Ktésibios [LMV92]. Au XVIII^{e} siècle, René-Antoine Ferchault de Réaumur a créé des dispositifs automatiques pour contrôler la température des incubateurs. Le dispositif était basé sur l'expansion thermique du mercure. L'élévation du mercure actionnait un flotteur ouvrant une vanne du four, permettant de contrôler la température à l'intérieur de l'incubateur.

Le contrôle-commande moderne s'est développé à partir des premiers régulateurs de moteurs à vapeur rotatifs, comme le régulateur à commande proportionnelle de James Watt en 1789. Les premières descriptions basées sur des équations différentielles ont rencontré des défis pour garantir la stabilité du système. Maxwell a établi que l'instabilité est liée à la fonction de transfert du système en 1868 [Max68]. En 1922, Nicholas Minorsky a publié la première formulation de la loi de commande appelée aujourd'hui PID (proportionnel-Intégral-Dérivé). Des stratégies de commande plus complexes ont émergé par la suite, comme la structure régulatrice RST. La structure mathématique du régulateur RST est une division polynomiale, permettant d'atteindre n'importe quel ordre [Ben96]. Un des principaux enjeux du contrôle-commande est le compromis entre performances et robustesse [Dua06]. Cette préoccupation sera prise en compte pour chaque analyse de stratégie de contrôle, tout comme la difficulté de mise en œuvre et la puissance de calcul nécessaire.

2.3.1.1 Commande PID

Le PID est une stratégie de contrôle qui consiste à réguler le système en appliquant des gains à l'erreur mesurée, la dérivée de l'erreur et l'intégrale de l'erreur. Il existe un consensus au sein de la communauté scientifique selon lequel presque tous les systèmes industriels, à l'exception de quelques systèmes précis spécialisés, sont réglés à l'aide d'un PID, même s'il n'y a pas à notre connaissance de publication récente permettant de retracer ou d'estimer la répartition exacte.

Cette stratégie est désormais largement étudiée, avec des sous-contrôleurs tels que l'adaptatif, la logique floue ou d'autres, comme cela est détaillé dans [Cro+05].

Les points forts de cette stratégie sont sa facilité de réglage avec la paramétrisation connue sous le nom de Ziegler-Nichols [ZN42]. La paramétrisation de Ziegler-Nichols permet de trouver les meilleurs paramètres pour un compromis entre performance et robustesse. Cependant, le PID n'apporte des résultats satisfaisants que jusqu'aux systèmes du deuxième ordre. Un autre avantage de cette stratégie est son faible coût de calcul : une multiplication pour le terme P, un décalage moyen et une multiplication pour le terme I, une soustraction et une multiplication pour le terme D suffisent.

Cette stratégie peut être appliquée pour manipuler des objets déformables, comme dans [KP12]. Dans cet article, le PID est l'une des deux stratégies proposées. Le PID n'a pas réussi à stabiliser le système.

2.3.1.2 Commande optimale

Les commandes optimales regroupent les stratégies de commande utilisant des stratégies mathématiques d'optimisation – c'est à dire minimiser une fonction de coût – appliquées à des système sous représentation d'état. La plus répandue est la commande H_2. La commande H_2 est l'application de la norme H_2 dans le cadre d'une commande LQG. La commande LQG est une extension de la commande LQ, où LQ signifie linéaire quadratique.

La stratégie LQ consiste à résoudre un problème d'optimisation en utilisant des critères mathématiques. Cette solution prend en compte la minimisation de l'erreur, le temps nécessaire pour résoudre le problème et l'action d'entrée, tout en fournissant un effet de stabilisation supérieur à celui du régulateur RST basé sur le placement des pôles [CC12]. Cette stratégie suppose que le système est linéaire, même si des extensions non linéaires existent.

LQG signifie linéaire quadratique gaussien. La partie "gaussien" provient de l'ajout d'un filtre de Kalman pour estimer certaines parties de notre état qui ne sont pas mesurables directement [Gov19].

En mettant la commande résultante sous une forme normée – avec pour norme la norme euclidienne de la fonction de transfert entre les perturbations d'entrée et la sortie du système, pondérée par la fréquence, nous obtenons la commande H_2. Cette stratégie fournit un résultat optimal, c'est à dire la solution d'un problème d'optimisation. Le résultat est le meilleur contrôleur possible pour un compromis performance/robustesse exprimé mathématiquement par le biais de matrices de poids. L'adéquation du résultat aux problèmes du monde réel n'est pas garantie : toute la difficulté est de régler correctement les critères mathématiques et le coût des actions pour répondre aux besoins d'une application.

Le coût de calcul de cette stratégie est plus élevé que celui du PID, car il implique des calculs matriciels. Le coût peut diminuer si les matrices d'évolution du système d'état sont invariantes dans le temps. Dans ce cas, certaines des matrices nécessaires peuvent être calculées hors ligne ou une seule fois, ce qui permet d'économiser de la puissance de calcul.

2.3.1.3 Commande H_∞

La commande H_∞ est une application d'une norme H_∞ à un problème similaire à celui décrit dans la section 2.3.1.2. Cette norme permet de trouver les performances optimales avec une marge souhaitée. Un tel problème peut être résolu à l'aide d'inégalités linéaires matricielles [BB08]. Il est également possible d'intégrer à la commande une fonction de façonnage, fonction permettant d'augmenter la performance dans une gamme de fréquences donnée tout en maintenant la robustesse en dehors de cette bande passante.

Par définition, cette stratégie est placée du côté de la robustesse dans le compromis performance/robustesse. Du point de vue computationnel, cela est légèrement plus coûteux que la stratégie décrite dans la section 2.3.1.2, car nous devons résoudre en ligne un problème d'optimisation sous contraintes.

Cette stratégie de commande dépasse de loin les performances de la commande par retour d'état en placement de pôles : elle est à l'optimum du compromis performance/robustesse formulé. Cependant, elle nécessite plus de calcul. Cette stratégie peut être appliquée à la manipulation d'objets déformables, comme décrit dans [KP12]. Dans cet article, la commande H_∞ est utilisée pour réaliser des tâches de manipulation indirecte de points de consigne. L'objet déformable a été modélisé à l'aide de l'approche FEM, décrite dans la section 2.1.1.3. Le régulateur H_∞ a pu réaliser avec précision les tâches de régulation sans nécessiter de réglage supplémentaire.

2.3.1.4 Model Predictive Control

La commande MPC (Model Predictive Control) est une extension de la stratégie H_2. La partie "prédictive" de la MPC provient de l'horizon de prédiction. Cette commande consiste à appliquer la stratégie H_2 et à la calculer sur plusieurs pas discrets, appelés horizon de contrôle, tout en modélisant les perturbations entrantes sur un horizon séparé appelé horizon de prédiction. Cette stratégie permet de rejeter les perturbations avant qu'elles ne se produisent, mais il est essentiel que les perturbations soient bien modélisées pour que cette stratégie soit efficace. Cela est particulièrement adapté aux perturbations cycliques [Cam+07].

Si les perturbations sont rejetées en utilisant cette stratégie, les résultats obtenus sont meilleurs qu'avec la stratégie H_2. Cependant, le coût en calcul est très élevé, car il est nécessaire de simuler et de calculer la commande pour plusieurs pas futurs, même si seul le calcul du prochain pas de temps est utilisé.

Il existe de nombreux exemples d'objets déformables dans la littérature, principalement dans le domaine de la robotique médicale :

- En 2017, Calli et al. [CD17] ont manipulé la position de cylindres et d'objets rectangulaires en utilisant uniquement la préhension par les doigts d'une main robotique, en comparant l'IBVS (cf. section 2.3.1.5) à une commande adaptative et à la MPC. La MPC a donné de meilleurs résultats en termes de précision et de temps nécessaire pour effectuer la tâche.
- La même année, Vrooijink et al. [Vro+17] ont utilisé la MPC pour compenser les battements du cœur lors de chirurgies mini-invasives.
- En 2018, Ouyang et al. [Ouy+18] ont utilisé un robot flexible à continuum pour contrôler la déformation d'un objet mou. La jacobienne de l'objet est estimé en ligne. La déformation de l'objet est régulée à l'aide de la MPC tout en évitant les obstacles.
- En 2019, Hyatt et al. [HWK19] ont modélisé la dynamique d'actionneurs souples à l'aide d'un réseau neuronal. Ils ont extrait de ce réseau neuronal une représentation d'état discrète linéarisée du système. Ils ont utilisé cette représentation pour développer une MPC.
- La même année, Shin et al. [Shi+19] ont effectué une manipulation de tissus avec des robots chirurgicaux. Ils ont utilisé une MPC basée sur l'apprentissage pour placer des points

spécifiques sur le tissu à des positions souhaitées. Leur modèle a appris le comportement non linéaire du tissu à partir de l'espace des images.

Bien que l'efficacité d'une telle stratégie soit indéniable, il n'est pas utile dans notre cas d'utilisation, car aucune perturbation ni évitement d'obstacles dynamiques n'est requis ni pour la tâche de soudage des pneus, ni pour les autres tâches du projet.

2.3.1.5 Asservissement visuel

Les stratégies d'asservissement visuel, ou Visual Servoing, consistent à contrôler le mouvement d'un robot dans l'espace image. Il existe deux principales stratégies : l'Image-Based Visual Servoing (IBVS) et le Position-Based Visual Servoing (PBVS).

Dans l'IBVS, des caractéristiques sont détectées dans l'espace de l'image. L'asservissement fonctionne en faisant correspondre ces caractéristiques avec des caractéristiques cibles, ce qui permet d'appliquer la stratégie de contrôle uniquement dans le plan 2D de l'image.

Dans le PBVS, la caméra est utilisée pour obtenir des positions en déprojetant les pixels suivis, les caractéristiques ou d'autres mesures d'image à l'aide des caractéristiques 3D des capteurs. Ces positions sont utilisées pour calculer une erreur de position dans l'espace 3D. La stratégie à préférer dépend du cas d'utilisation [CH06] [CH07].

2.3.1.6 Commande adaptative

La commande adaptative utilise un contrôleur adaptatif qui se met à jour en temps réel pour corriger des paramètres inconnus et variables [LLM+98]. Cette stratégie est compatible avec les précédentes. Dans la littérature, cela est souvent exprimé comme un apprentissage du contrôleur.

Ces stratégies sont basées sur des techniques d'identification, améliorées grâce aux progrès de l'apprentissage automatique et des réseaux de neurones, pour créer une logique floue neurale (neuro-fuzzy logic) [Bou+14].

2.3.1.7 Stabilité

Une préoccupation majeure de l'ingénierie des systèmes est la stabilité. La définition de stabilité utilisée pour la commande H_2 est la stabilité de Lyapunov, détaillée dans le livre [Lya92]. Cela vise à garantir la stabilité du système mécanique de commande sur des points d'équilibre potentiellement instables. Dans notre cas d'utilisation, l'instabilité de forme est causée par des flexions inattendues.

2.3.1.8 Conclusion

La précision du contrôle dépendra forcément des informations fournies par les capteurs [SKM19b]. Dans le cas des objets déformables, un modèle de déformation approximatif peut être compensé par une perception de forme de haute précision.

La table 2.7 résume les avantages et inconvénients des diverses commandes exposées dans cet état de l'art. Dans le cadre du problème industriel à résoudre, la Jacobienne à le rôle de modèle pour les commandes par retour d'état. Notre stratégie de commande sera une extension d'un asservissement visuel, Le problème ne semble pas nécessiter des performances de robustesse ou de rejet de perturbations particulièrement avancées - d'où l'inadéquation de la commande H_∞. De même, l'absence de perturbation à modéliser rend caduque l'application de la stratégie MPC. Les stratégies Adaptatives permettent de commander un système avec un modèle approximatif, ou changeant ; cependant nous pensons que l'utilisation de Jacobienne permet d'avoir un modèle

suffisant à l'asservissement de forme. C'est pourquoi notre intérêt s'est porté vers la stratégie de commande optimale H_2 dans le Chapitre 3.

TABLE 2.7: Table de comparaison des lois de commande

Stratégie	Avantages	Inconvénients
PID	Simple et intuitive. Légere en calcul.	Stratégie limitée.
H_2	Optimale mathématiquement.	Optimale selon des critères définis par l'utilisateur. Nécessite de résoudre un problème d'optimisation en ligne.
H_∞	Meilleure performance pour une robustesse garantie.	Nécessite de résoudre un problème d'optimisation encore plus difficile en ligne.
MPC	Peut anticiper et corriger les perturbations avant qu'elles ne se produisent.	Nécessite de résoudre plusieurs fois le problème d'optimisation H_2 à chaque instant. Nécessite un modèle des perturbations.
Adaptatif	Excellente avec les non-linéarités ou les systèmes changeants. Peut être combinée avec l'une des stratégies précédentes.	Incertitude de robustesse.

2.3.2 Asservissement de forme

L'asservissement de forme est un sous-problème de la mise en forme, consistant à déformer un objet pour atteindre une forme donnée. Il s'agit des stratégies de contrôle en boucle fermée permettant d'amener un objet déformable d'une forme initiale quelconque à une forme finale donnée. Il ne s'agit pas de la seule problématique impliquant les objets déformables : on peut citer par exemple le packaging, l'insertion. Les précurseurs du domaine viennent de l'infographie, en particulier la stratégie sans modèle du papier précédemment cité [Ber13], dont les résultats sont visibles dans la Fig. 2.8.

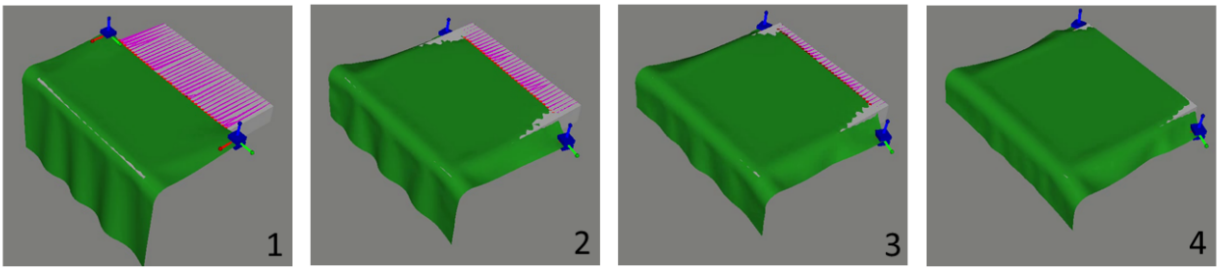

FIGURE 2.8: Simulation de la mise en place d'une nappe

Extrait du papier [Ber13]

Le terme "asservissement de forme" vient du papier [NAL17], un autre précurseur, qui s'intéresse au contour de son objet déformable et contrôle les coefficients de Fourier de la fonction

traçant la forme courante pour l'emmener vers une forme cible.

2.3.2.1 Asservissement de forme en infographie

Animer un personnage ou un objet déformable à partir d'une forme initiale donnée jusqu'à une forme cible, en respectant des contraintes de déformations et de mouvements, est un problème largement étudié dans le domaine de l'infographie, comme discuté plus haut pour la stratégie présenté dans [Ber13]. L'infographie est proche de nos problématique sur les aspects de modélisation et de commande : l'objet est simulé avec un modèle de déformation, et les formes prises par l'objet peuvent être issus d'algorithmes en boucle fermée cherchant à atteindre une forme finale. Par exemple, dans [DSB99], les auteurs ont simulé les interactions entre un morceau de tissu et diverses formes. Ce papier est précurseur dans le domaine de l'asservissement de forme car propose des approches de modèle et stratégies de contrôle permettant de fermer la boucle. Dans [BSP09] et [Sch+14], les auteurs ont utilisé un contrôle optimal pour générer des contraintes internes dans des corps déformables, respectivement un dinosaure en plastique et un personnage en forme de X illustré Fig. 2.9, afin de créer des mouvements réalistes. L'objet est modélisé et les commandes générées permettent de créer une succession d'étape intermédiaire réaliste entre une forme initiale et une forme finale en prenant en compte l'état interne de l'objet, soit un objectif fortement proche de celui de nos travaux de recherche. Le monde réel rajoute des difficultés : tout d'abord, la différence entre le modèle et la réalité. Dans les mondes virtuels des infographiste, le modèle de la commande est le même que le moteur de calcul de déformation de l'univers. Dans la réalité, aucun modèle n'est parfait. Ensuite, le problème de perception : la réalité nécessite un moyen pour reconstruire l'état de l'objet, accessible directement en simulation. Enfin, les contraintes de vitesse. L'infographie peut se permettre de prendre du temps pour générer ses rendus.

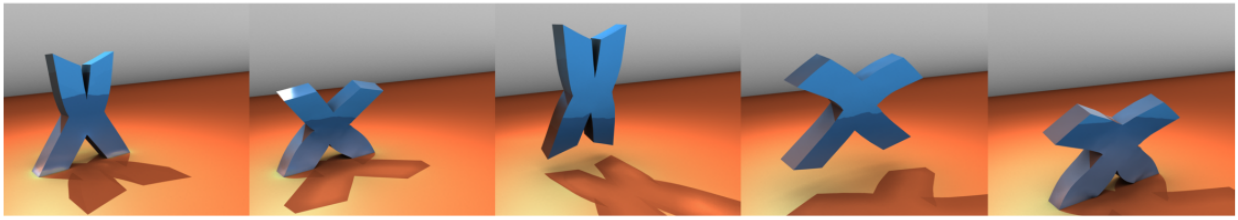

FIGURE 2.9: Simulation sous contrainte de l'animation d'un personnage en forme de X

Extrait du papier [Sch+14]

2.3.2.2 Asservissement de forme en robotique

Les problématiques d'asservissement de forme sont fortement dépendantes des propriétés de l'objet déformable manipulé. Nous présenterons donc les stratégies adaptées aux différents types d'objets présentés dans la revue [San+18].

Asservissement de forme d'objets déformables linéique :

Un objet linéique est un objet dont la seule caractéristique pertinente est sa longueur. Le problème peut être résolu avec des stratégies de commandes spécifique à l'objet déformable :

Manipulation de poutres : Si la résistance à la compression de l'objet linéique est élevée, notre objet peut être assimilé à une poutre. Dans la littérature, plusieurs stratégies ont été appliquées à de telles poutres. Dans [Agh+22], la poutre est manipulée en appliquant As-Similar-As-Possible, un modèle proche d'As-Rigid-As-Possible. Dans cette application, la poutre modélise une branche de plante. Dans [BM14], un travail mathématique d'analyse géométrique sur les

configurations d'équilibre a été réalisé sur une telle barre, réduisant l'ordre des formes possibles qu'un fil flexible peut prendre grâce à la réduction de Lie-Poisson. Ce papier démontre que les formes possibles suivent un manifold lisse de dimension finie qui peut être paramétré par un seul graphique. Cela ouvre la question d'à quel point un problème de manipulation d'objet déformable est sous-actionné.

Solutions basées sur l'équation de la chaînette : Si la résistance à la compression de l'objet linéique est faible ou négligeable, il peut quand même se maintenir entre des supports. Dans ce cas, la courbe que l'objet crée lorsqu'il est soumis à son propre poids suit, comme indiqué précédemment, l'équation de la chaînette.

Cette équation a été adaptée pour le contrôle robotique des objets déformables. Dans [Fil+22], l'équation de la chaînette est utilisée pour déduire la tension dans des bandes de caoutchouc, permettant de minimiser la tension dans l'objet déformable manipulé en contrôlant la longueur pendante à l'aide d'une bobine motorisée. Ce set-up est montré dans la Fig. 2.10. Ces travaux ne sont pas des travaux d'asservissement de forme stricto sensu, car la forme de l'objet par le positionnement de l'effecteur est uniquement un moyen d'obtenir et de modéliser la tension actuelle et de l'asservir, ce qui montre les possibilités ouvertes par la résolution des problématiques d'asservissement de forme. Dans [Dru+22], plusieurs robots sous-marins sont liés les uns aux autres par un câble prenant la forme d'une chaînette. Cela permet d'estimer la position des robots les uns par rapport aux autres et d'utiliser l'affaissement de la chaînette comme une entrée de contrôle, asservissant ainsi un objet déformable.

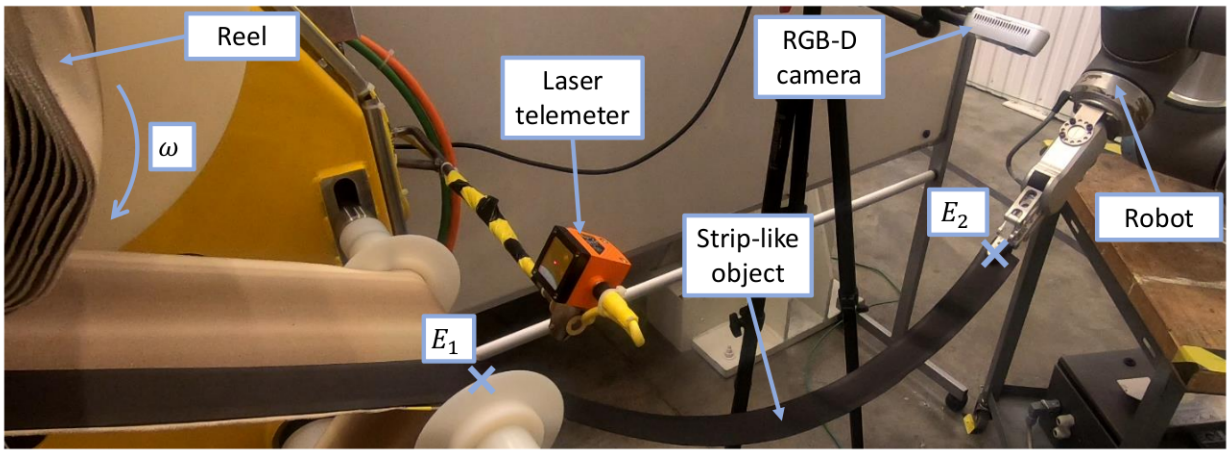

FIGURE 2.10: Asservissement de la tension d'une bande de gomme grâce à l'équation du caténaire

Extrait du papier [Fil+22]. Le cas d'usage est proche de notre cas d'usage privilégié.

Manipulation de corde : Un cas bien étudié est la manipulation de corde. Des travaux récents se concentrent sur les stratégies de routage, comme [KBS22] qui réduit l'ordre du problème en représentant le plan de travail comme un graphe ; les stratégies d'asservissement de forme comme [YSS22] qui utilise un réseau de neurones pour apprendre le modèle de l'objet manipulé afin de lui appliquer une stratégie de contrôle MPC ; les stratégies d'asservissement de tension comme [SLK22] qui incorpore les retours provenant des capteurs de force embarqués dans les préhenseurs.

Une autre application possible est liée à la robotique continue, comme dans [BWV22], où le problème est enrichi avec le retour d'informations d'un capteur intégré, ce qui permet un asservissement et une reconstruction plus précise que le problème classique d'asservissement de forme où l'objet dont la forme est à asservir est saisi par le manipulateur.

Asservissement de forme d'objets déformables surfaciques :

L'un des premiers travaux sur la manipulation d'objets déformables est la stratégie décrite dans [Ber13], qui se trouve être appliquée sur un objet suivant la formulation surfacique. Cette stratégie se base sur un contrôleur jacobien proportionnel, avec une jacobienne calculée en utilisant le principe de rigidité décroissante. Cette stratégie a été implémentée en simulation uniquement, souffrant du problème de l'implémentation de la perception de forme.

Les travaux utilisant l'approche As-Rigid-As-Possible et un contrôleur proportionnel, désigné par la suite sous le nom ARAP-SS, pour ARAP-Shape Servoing, présentés dans le papier [SB+22], développés dans le cadre du projet SoftManBot, portés dans l'architecture proposée sont présentés en détail dans la section 4.4.4. Ces travaux ont été prolongés en utilisant en simulation un contrôle d'une formation de robot manipulant un objet déformable évitant des obstacles [Her+22].

Asservissement de forme d'objets déformables volumiques :

Nous considérons dans cette section l'asservissement dans un espace 3D d'objets volumiques. Ces restrictions complexifient le challenge, le rendant encore plus lourd en calcul. Les strategies d'asservissement fonctionnant sous ces hypothèses se focalisent soit sur une partie de l'objet uniquement dans le cas de contrôle jacobien proportionnel [LKM20], soit ne parvient pas à fonctionner en temps réel [ZPC21] [Due+18] : ces stratégies utilisent une modélisation complète de l'objet et appliquent des commandes en simulation pour obtenir une forme cible, comme montré Fig. 2.11. La puissance de calcul nécessaire à ces stratégies ne permet pas un asservissement visuel en boucle fermée. Une comparaison des stratégies de contrôle d'objets déformables est détaillée section 3.1, et résumée dans le tableau 3.1

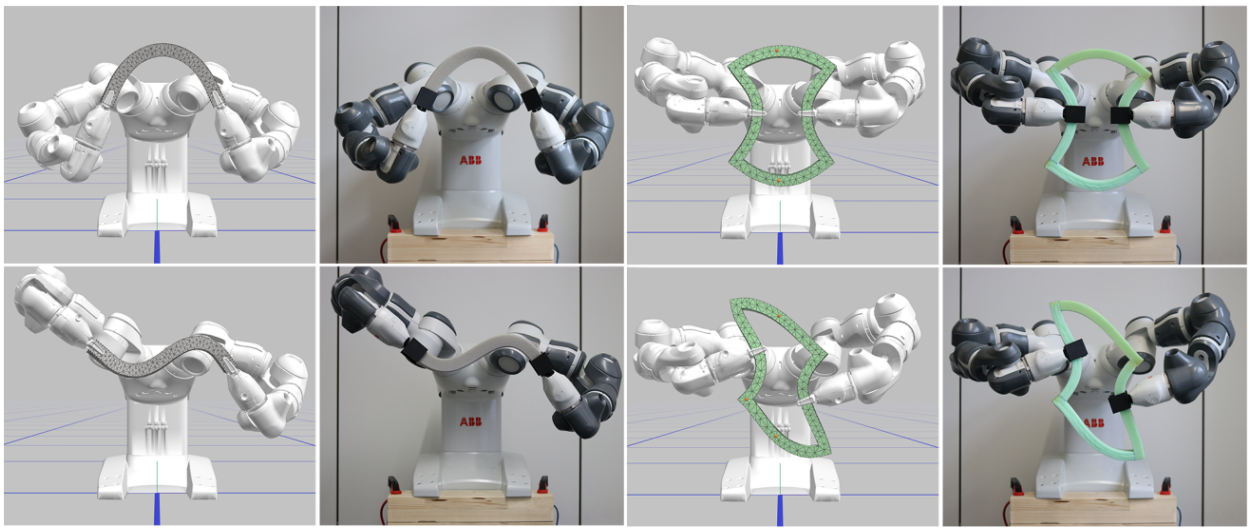

FIGURE 2.11: Asservissement d'un objet volumique

Extrait du papier [Due+18]

Analyse critique de l'état de l'art relativement à la problématique de thèse

Dans nos travaux, la problématique industrielle d'asservissement de bande de roulement nécessite :

— Une stratégie d'asservissement de forme adaptée à un objet surfacique ou volumique.
— Un asservissement de forme avec un temps de cycle de l'ordre de la centaine de millisecondes.
— Une précision de l'ordre du millimètre.

— Un asservissement prenant en compte la trajectoire de déformation car l'objet est collant.
Cependant, la tâche peut se dispenser d'une simulation de la dynamique de l'objet manipulé. Dans l'état de l'art, l'asservissement de forme ARAP-SS est le plus prometteur, associé à une stratégie de commande plus avancé que le contrôleur proportionnel.

2.4 Architectures logicielles pour systèmes robotiques

Une architecture logicielle est définie, d'après la norme IEEE [2], par "*la pratique recommandée comme l'organisation fondamentale d'un système, incarnée dans ses composants, leurs relations les uns avec les autres et avec l'environnement, ainsi que les principes régissant sa conception et son évolution*".

La robotique moderne a été fortement structurée par ROS (Robot Operating System) [Qui+09], qui s'est imposé comme un standard de facto. ROS est un middleware de robotique décentralisé où les effecteurs et capteurs ont des programmes dédiés appelés "nœuds" communiquant dans un même réseau. Le middleware propose des interfaces sous forme de messages, de services et d'actions entre les nœuds, ainsi que des outils permettant d'afficher les informations transitant entre ces nœuds, de les enregistrer et de les rejouer. Les concepteurs d'algorithmes ainsi que les constructeurs de capteurs et d'actionneurs peuvent donc regrouper toutes les interfaces de leurs composants dans un même "Package" contenant les nœuds développés. 13 distributions de ROS ont été développées entre 2010 et 2020 pour la version 1, et 9 distributions de ROS2 ont été développées entre Décembre 2017 et Juin 2023. La compatibilité des packages au sein de distributions différentes d'une même version n'est pas automatique, d'où l'intérêt de partager les sources avec la communauté des développeurs.

Un des avantages de ROS2 est la possibilité d'intégrer des fonctionnalités respectant le temps réel [Fis+19], qui n'est pas garanti avec ROS1.

La plupart des industriels sont confrontés à des problèmes spécifiques et ont tendance à développer une couche d'architecture au-dessus de ROS. Par exemple, dans [Nam+20], l'architecture fonctionne uniquement pour les solutions basées sur les réseaux de neurones, tandis que la solution développée dans [SO+20] est spécifiquement conçue pour un essaim de robots. L'architecture proposée dans [Sim+21] convient aux cellules de travail reconfigurables à faible volume de production.

L'idée de procédés plus globaux est largement présente dans la communauté robotique. En 2006, le framework ORCA [Bro+05] proposait le paradigme de Robotique orientée composants : en reprenant le paradigme de composants logiciels, on peut séparer une application robotique à l'échelle de système en composants, unités mêlant logiciel et matériel formant des modules interchangeables. On obtient au final un système bien plus maintenable et expansible grâce à ce paradigme.

Plus récemment, plusieurs solutions sont proches de nos problématiques :

•*RobMoSys* [3] est principalement orienté pour l'industrie. Cette solution propose un cadre complet, mettant l'emphase sur la modularité et la modélisation – proposant outil et formalisme correspondant. L'échelle des applications construites avec ce framework dépasse le système : le nombre d'interactions logicielles et matérielles typiquement prises en charge atteint le statut d'écosystème.

•*skiROS* [Rov+17] est un autre framework qui permet de diviser les fonctionnalités de haut niveau en tâches, constituées de compétences elles-mêmes constituées de primitives. Un non-expert peut assembler des compétences pour atteindre des objectifs, une fonctionnalité utile pour la planification. Le framework est la propriété de l'entreprise Riact. Il est gratuit à utiliser, mais non

2. https ://standards.ieee.org/ieee/1471/2187/
3. https ://robmosys.eu/

commercialisable.

•*PlanSys2* [Mar+21] est un framework équivalent à skiROS, pour ROS2, à la différence majeure de la propriété : PlanSys2 est un projet open-source. De plus PlanSys2 propose l'implémentation de solutions de planification réactive permettant d'adapter la prise de décision à l'évolution de l'environnement.

Chapitre 3

Contribution : Optimal Shape Servoing

Ce chapitre propose une stratégie d'asservissement de forme innovante, Optimal Shape Servoing (OSS). Le but de la stratégie est de parvenir à intégrer des contraintes implicites de trajectoire de déformation à un contrôleur de forme. Ces contraintes sont incluses par le biais de matrices de poids, permettant d'influencer les formes intermédiaires prises par l'objet déformable au cours de la tâche.

Ce chapitre développe la théorie de cette stratégie d'asservissement de forme, des simulations et trois cas d'application : assemblage de gomme, mise en boîte d'un poster, apprentissage par démonstration.

— Le premier cas d'application permet de mettre en avant la praticité de la stratégie dans le cas d'usage privilégié.
— Le second cas d'application a été choisi pour mettre en avant la possibilité qu'offre la stratégie d'asservissement de forme de décorréler l'erreur de position et l'erreur de forme.
— Le dernier cas d'application présente une technique permettant le réglage des coefficients à partir des données obtenues en effectuant la tâche.

3.1 Positionnement relativement à l'état de l'art

La manipulation robotique d'objets déformables est un sujet actif de recherche, comme présenté dans la section 2.3.2. Elle a un large éventail d'applications au-delà de l'industrie, par exemple dans la robotique de service, l'assistance chirurgicale [Shi+19] ou encore l'emballage alimentaire [GS14]. Le principal défi de la manipulation d'objets déformables est de générer les commandes appropriées pour contrôler la forme de l'objet. C'est un problème ouvert connu sous le nom de problème d' asservissement de forme [NAL17 ; San+18 ; NVP18].

Bien qu'il existe des stratégies récentes d'asservissement de forme [Han+21 ; LKM20 ; SB+22], elles ne sont pas en mesure de satisfaire les contraintes de convergence pour atteindre la forme souhaitée. La manière de converger vers la forme finale est particulièrement critique pour accomplir des tâches d'assemblage précises, telles que l'assemblage de surfaces déformables adhésives. Cette section a pour but d'analyser les solutions proposées dans l'état de l'art, et de les positionner vis-à-vis d'OSS.

Dans [KP12] l'objet est surfacique. Les auteurs ont modélisé la déformation de l'objet à l'aide de la FEM. Ils ont observé l'objet à l'aide d'une caméra stéréo, et ont contrôlé deux points de l'objet, qui étaient des LED suivies par la caméra stéréo, dans le plan. Ils ont utilisé une stratégie d'asservissement en boucle fermée avec un contrôleur H^{∞}. Celui ci ne prend pas en compte de contraintes de convergence. Cependant, il incluait des contraintes de mise en forme de la boucle. Les auteurs n'ont pas prouvé la stabilité de la loi de commande.

Dans [NAL17], l'objet est volumique. Les auteurs ont observé l'objet à l'aide d'une caméra RGB monoculaire. Ils ont extrait les coefficients de Fourier du contour de l'objet et ont contrôlé le contour en utilisant ces coefficients de Fourier dans l'image. Les auteurs ont utilisé un contrôleur d'asservissement visuel proportionnel [CH06] avec une estimation en ligne de la jacobienne. Les auteurs n'ont pris en compte aucune contrainte de convergence et ils n'ont pas prouvé la stabilité de la loi de commande.

Dans [LKM20], l'objet est surfacique ou volumique. Les auteurs ont prédit la déformation de l'objet à l'aide d'une jacobienne estimée en ligne sans modèle : la jacobienne est calculée grâce à la technique des moindres carrés, pondéré sur une fenêtre glissante, en fonction des manipulations précédentes. Un nuage de points d'une région d'intérêt de l'objet est observé à l'aide d'une caméra RGB-D et contrôlé dans l'espace 3D. Les auteurs ont utilisé une stratégie en boucle fermée avec un contrôleur par descente de gradient proportionnelle. Ils ont également utilisé une fenêtre glissante pondérée avec un critère d'oubli. Aucune contrainte de convergence n'est prise en compte. La stabilité de la loi de commande a été prouvée.

Dans [ZPC21], l'objet est volumique. Les auteurs ont modélisé la déformation de l'objet en utilisant la méthode de Newton avec la programmation dynamique différentielle (DDP). Ils ont modélisé la géométrie de l'objet avec un maillage tétraédrique, et n'ont utilisé aucun capteur de vision, car l'asservissement était en boucle ouverte. Les auteurs ont contrôlé l'ensemble de la forme de l'objet dans l'espace 3D. Ils n'ont pris en compte aucune contrainte de convergence, mais l'objet suit une trajectoire générée en prenant en compte la dynamique pour accomplir les tâches souhaitées.

Dans [Han+21], l'objet est linéique, surfacique ou volumique. Les auteurs ont prédit la déformation de l'objet à l'aide d'une jacobienne estimée en ligne à l'aide de la technique d'approximation de fonction [Ebe+19]. L'objet a été observé à l'aide d'une caméra RGB monoculaire et trois points de l'objet dans l'image sont contrôlés. Les auteurs ont utilisé une stratégie d'asservissement de forme en boucle fermée avec un contrôleur adaptatif. Ils n'ont pris en compte aucune contrainte de convergence, mais une trajectoire prédéterminée est suivie. Les auteurs ont inclus des contraintes de force virtuelle pour améliorer la manipulabilité. La stabilité de la loi de commande a été prouvée.

Dans [SB+22], l'objet est surfacique. Les auteurs ont modélisé la déformation de l'objet en utilisant l'approche As-Rigid-As-Possible (ARAP) [SA07]. Ils ont observé la forme de l'objet avec un algorithme Shape-from-Template (SfT) [Bar+15 ; ÖB17] en utilisant une caméra RGB monoculaire. L'ensemble de la forme de l'objet est contrôlé dans l'espace 3D. Les auteurs ont utilisé une stratégie en boucle fermée avec un contrôleur proportionnel. Ils n'ont pris en compte aucune contrainte de convergence. La stabilité de la loi de commande n'a pas été montrée car la stratégie était fortement sous-actionnée.

TABLE 3.1: Comparaison de l'état de l'art des stratégies dédiées à la manipulation d'objets déformables

	Objet	**Capteur de vision**	**Variables d'asservissement**	**Espace de contrôle**	**Loi de commande**	**Trajectoire prédéfinie adaptée à la tâche**	**Contraintes de trajectoire adaptées à la tâche**
Kinio, 2012	Surfacique	Stéréo	2 points	Plan	H^∞	Non	Non
Navarro-Alarcon, 2018	Volumique	RGB	Contour	Image	Proportionnel	Non	Non
Lagneau, 2020	Volumique	RGB-D	Nuage de points	3D	Proportionnel	Non	Non
Zimmerman, 2021	Volumique	Aucun	Forme entière	3D	Boucle ouverte	Oui	Non
Han, 2021	Volumique	RGB	3 points	Image	Adaptatif	Oui	Non
Shetab-Bushehri, 2022	Surfacique	RGB	Forme entière	3D	Proportionnel	Non	Non
OSS, 2022	Surfacique (Volumique ou Linéique)	RGB (RGB-D ou Stéréo)	Forme entière	3D	Optimal	Non	Oui

3.2 Présentation de la stratégie Optimal Shape Servoing

Pour répondre aux problématiques de manipulations d'objets déformables présentées dans le chapitre 1, nous avons choisi de traiter le problème grâce à une représentation d'état : nous voulons contrôler les variables internes de l'objet, définissant sa forme. Pour introduire les notions de trajectoire de déformation, une stratégie de contrôle permettant de contrôler la convergence semble adaptée. Le problème ne présente pas de perturbation anticipable : les stratégies de type MPC ne sont donc pas adaptées. Nous savons que le problème est solvable avec des stratégies linéarisé au point de fonctionnement, comme détaillé dans la section 4.4.4. C'est pourquoi nous avons choisi une commande inspirée des stratégies LQG, permettant d'obtenir une commande optimale - minimisant une fonction de coût, permettant de pondérer le coût de l'état avec une matrice nommée Q et le coût de la commande avec une matrice nommée R.

Dans la stratégie `OSS`, nous représentons la forme de l'objet sous forme d'un maillage composé de nœuds. Chaque nœud est relié à au moins deux autres par des triangles. Ensuite, nous utilisons une jacobienne de déformation pour prédire les variations de forme de l'objet en fonction des petits déplacements des effecteurs dans des conditions quasi-statiques. Nous écrivons l'équation d'état discrète du système de la manière suivante :

$$x(k+1) = x(k) + J(k)\,u(k)\,dt \tag{3.1}$$

où $k \in \mathbb{N}$ est le temps discrétisé, $dt \in \mathbb{R}$ est l'intervalle de temps, $x \in \mathbb{R}^{3n}$ est le vecteur d'état formé par les coordonnées des points nodaux, et n est le nombre de nœuds dans le maillage de l'objet. $u \in \mathbb{R}^{6m}$ représente la commande sous forme d'un vecteur empilé des vitesses des robots, où m est le nombre de robots. $J \in \mathbb{R}^{3n\times 6m}$ est la jacobienne de déformation.

$$x = \begin{pmatrix} x_1 \\ y_1 \\ z_1 \\ x_2 \\ y_2 \\ z_2 \\ \vdots \end{pmatrix}_{3n\times 1} \quad \text{and} \quad u = \begin{pmatrix} v_{x_1} \\ v_{y_1} \\ v_{z_1} \\ w_{x_1} \\ w_{y_1} \\ w_{z_1} \\ \vdots \end{pmatrix}_{6m\times 1} \tag{3.2}$$

La jacobienne de déformation J dépend d'un modèle de déformation représentant le comportement de l'objet. Elle prédit comment la forme de l'objet évoluera sous l'action des robots. Elle peut être calculée analytiquement ou estimée en ligne. La méthode de perception permettant d'obtenir l'estimation de la forme actuelle $x(k)$ ainsi que l'approche permettant de calculer la Jacobienne de déformation J importent peu pour cette commande. Pour les expériences, nous utilisons l'approche ARAP (cf. section 4.4.4) pour la modélisation et la méthode ROBUSfT (cf. section 4.4.3) pour la perception.

Nous définissons le vecteur d'erreur de forme $e \in \mathbb{R}^{3n}$ comme :

$$e(k) = x(k) - x_d(k) \tag{3.3}$$

où $x_d \in \mathbb{R}^{3n}$ est la forme désirée.

En appliquant la théorie du contrôle optimal [AM07], nous écrivons le coût global E en fonction du coût de l'erreur de forme et du coût de la commande de la manière suivante :

$$E = \sum_{k=1}^{t_h} \Big(e^\top(k)\,Q(k)\,e(k) + u^\top(k)\,R(k)\,u(k) \Big) \tag{3.4}$$

où $t_h \in \mathbb{N}$ est l'horizon temporel de notre commande. $Q(k) \in \mathbb{R}^{3n\times 3n}$ et $R(k) \in \mathbb{R}^{6m\times 6m}$ pondèrent respectivement le coût de l'erreur de forme et le coût de la commande. La commande u minimisant E est écrite comme suit :

$$u(k) = -K(k)\, e(k) \tag{3.5}$$

avec $K(k) \in \mathbb{R}^{6m\times 3n}$ défini comme suit :

$$K(k) = [G^\top(k)\, S(k+1)\, G(k) + R(k+1)]^{-1}\, G^\top(k)\, S(k+1) \tag{3.6}$$

où $G(k) = dt.J(k)$, d'après l'équation d'état, et $S(k) \in \mathbb{R}^{3n\times 3n}$ est défini comme suit :

$$\begin{cases} S(k) = S(k+1)\,[1 - G(k)\,K(k)] + Q(k) \\ S(t_h) = Q(t_h) \end{cases} \tag{3.7}$$

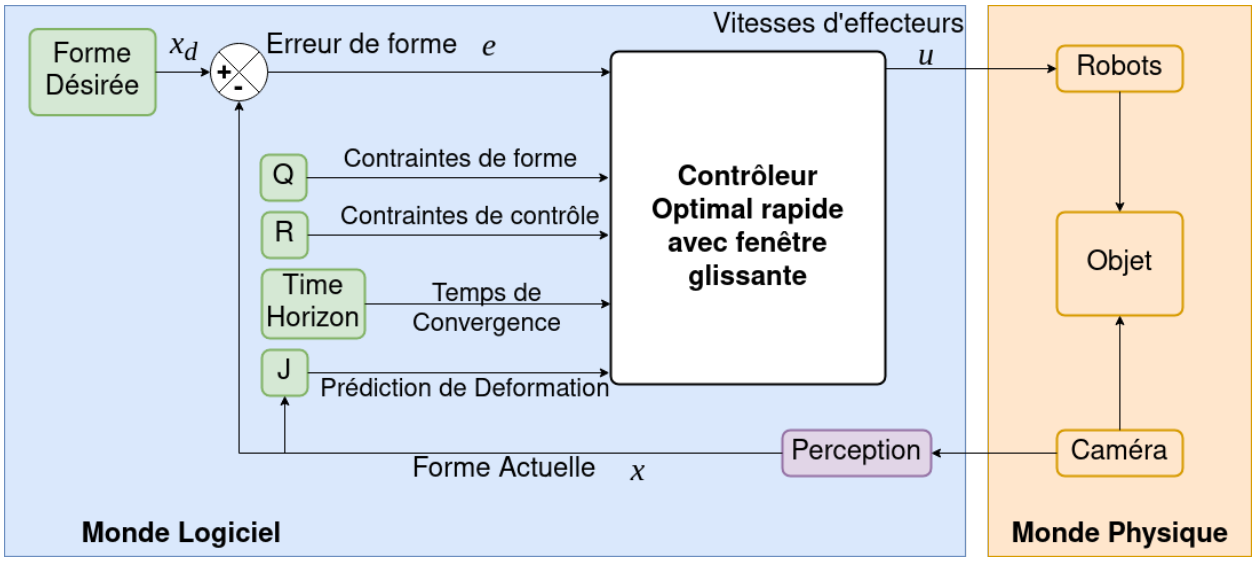

FIGURE 3.1: Schéma bloc d'OSS.

3.2.1 Synthèse des paramètres de commande

Cette section présente les différents paramètres ajustables, ainsi que leur impact sur la commande générée. Ces paramètres sont les matrices Q et R, ainsi que l'horizon temporel de la tâche.

Matrice Q pour les contraintes de forme L'objectif de la matrice Q n'est pas d'établir l'atteinte des formes intermédiaires, mais de déterminer la vitesse à laquelle différentes régions de l'objet convergent vers la forme finale désirée. Elle peut être constante ou varier avec le temps. Cette matrice nous permet, grâce au choix de ses coefficients, de donner plus ou moins d'impact à l'erreur générée par des états différents. Le choix de ces paramètres est complexe et multifactoriel : le choix doit prendre en compte la dynamique du système, les objectifs de contrôle, le compromis performance/robustesse, et bien sûr l'adéquation à la réalité. Nous proposons une formalisation donnant un sens physique à ce choix : donner un poids aux nœuds. Nous pouvons imaginer le coût comme un poids total, dépendant de l'erreur de chaque nœud. Rendre certains nœuds plus lourds signifie que leur erreur pèsera relativement plus que celle des nœuds plus légers. Cette formalisation est la suivante : Si nous fixons Q comme une matrice diagonale, le coût de l'erreur d'état E_e, définie comme

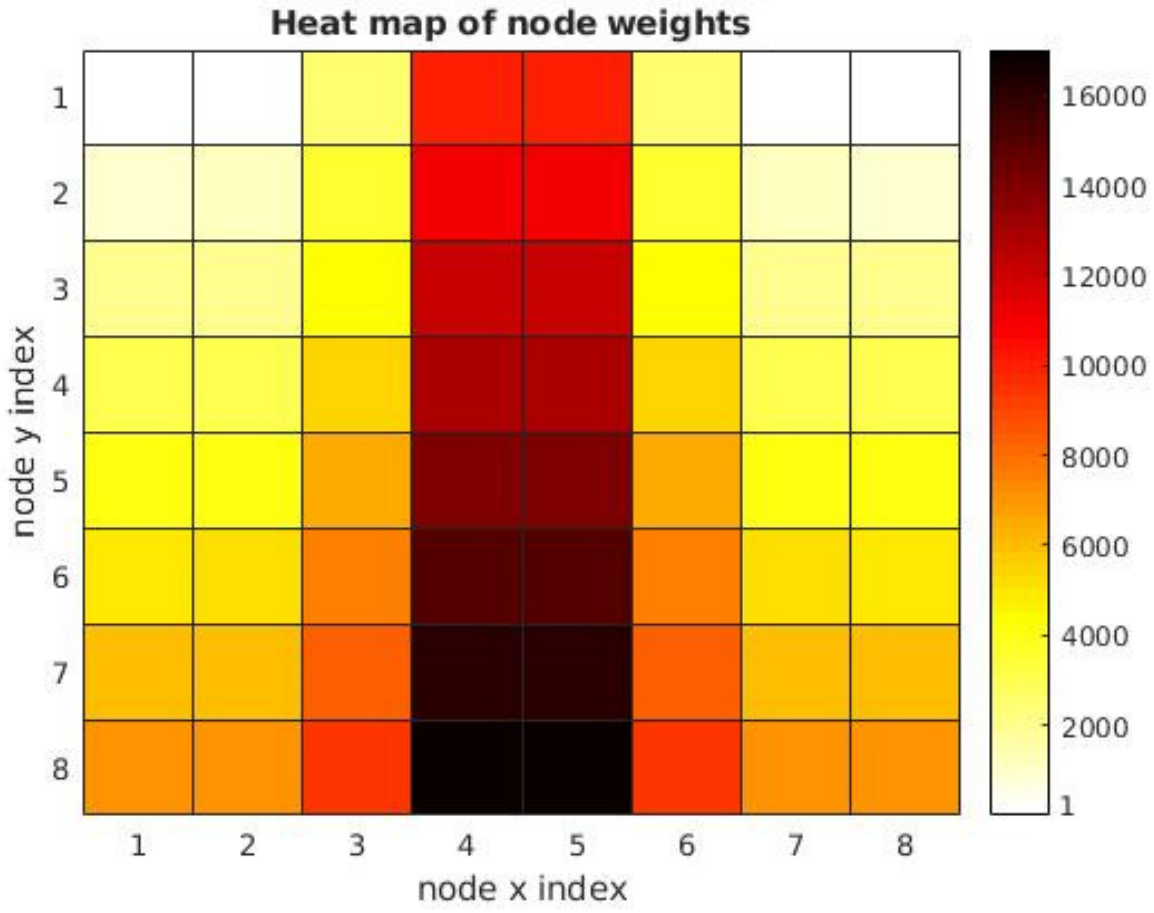

FIGURE 3.2: Représentation sous forme de "Carte thermique" des valeurs de la matrice Q sur les nœuds du maillage.

$$E_e = (x^T(t) - x_d^T(t))\, Q(t)\, (x(t) - x_d(t)) \tag{3.8}$$

devient

$$E_e = \begin{pmatrix} e_{x1} & e_{y1} & e_{z1} & \dots \end{pmatrix} \begin{pmatrix} Q_1 & 0 & 0 & 0 & 0 & 0 \dots \\ 0 & Q_2 & 0 & 0 & 0 & 0 \dots \\ 0 & 0 & Q_3 & 0 & 0 & 0 \dots \\ 0 & 0 & 0 & Q_4 & 0 & 0 \dots \\ 0 & 0 & 0 & 0 & Q_5 & 0 \dots \\ 0 & 0 & 0 & 0 & 0 & Q_6 \dots \\ & & & \dots & & \end{pmatrix} \begin{pmatrix} e_{x1} \\ e_{y1} \\ e_{z1} \\ e_{x2} \\ e_{y2} \\ e_{z2} \\ \dots \end{pmatrix} \tag{3.9}$$

avec e_{xi} (respectivement e_{yi} et e_{zi}) égal à la différence entre les positions actuelles et cibles sur l'axe x (respectivement y et z) du nœud d'indice i.

$$E_e = \sum_{k=0}^{n} (Q_{3k}.e_{xk}^2 + Q_{3k+1}.e_{yk}^2 + Q_{3k+2}.e_{zk}^2) \tag{3.10}$$

Nous avons remarqué que si Q est une matrice identité, nous pouvons donner une signification physique directe au coût de l'état. L'équation précédente devient alors

$$E_e = \sum_{k=0}^{n} ((x_k - x_{dk})^2 + (y_k - y_{dk})^2 + (z_k - z_{dk})^2) \tag{3.11}$$

L'équation (3.11) signifie que l'erreur de l'état est le carré de la distance entre la position actuelle de chaque nœud et la cible. Ensuite, si nous fixons Q comme une matrice diagonale telle que

$$\forall i < n, Q(3i,3i) = Q(3i+1,3i+1) = Q(3i+2,3i+2) \tag{3.12}$$

ce qui signifie qu'aucune direction n'est priorisée par rapport aux autres, le coût de l'état devient

$$E_e = \sum_{k=0}^{n} Q_k.((x_k - x_{dk})^2 + (y_k - y_{dk})^2 + (z_k - z_{dk})^2) \tag{3.13}$$

On peut attribuer des poids différents influençant directement le poids de l'erreur du nœud k en modifiant le coefficient Q_k. Les valeurs doivent être choisies en fonction de la tâche à accomplir. Plus de détails sur ces tâches et les valeurs peuvent être trouvés dans la section 3.4.2 et la section 3.4.3. Les paramètres des matrices de poids peuvent également être appris par démonstration, comme montré dans la section 3.4.4.

Matrice R pour les contraintes de contrôle : Cette matrice, représentant le coût de la commande dans l'équation 3.4 et impactant le gain matriciel calculé dans l'équation 3.6 permet de donner la priorité à certains DDL rotationnels, translationnels ou n'importe quelle combinaison de ces derniers aux effecteurs des robots. Par exemple, on peut donner la priorité aux DDL translationnels lors d'un asservissement de forme de matériau souple, car les déformations sont principalement dominées par la gravité et les effets des DDL rotationnels restent relativement locaux. La matrice R peut être constante ou variable dans le temps.

Horizon temporel pour le temps de convergence de la tâche : Le coût global E dans l'équation (3.4), qui inclut l'erreur de forme, est minimisé sur l'horizon temporel. Ce dernier est un paramètre de la commande : l'horizon temporel représente la vitesse à laquelle la tâche sera terminée (c'est-à-dire le temps de stabilisation). Imposer un temps de stabilisation souhaité pour une tâche complexe telle que l'asservissement de forme n'est pas trivial avec les schémas de contrôle conventionnels (par exemple, PID). Du point de vue industriel, il s'agit d'une propriété très utile.

3.3 Représentation algorithmique de la stratégie `OSS`

La stratégie d'asservissement de forme `OSS` est détaillée dans l'algorithme 1. Cet algorithme est un algorithme de commande optimale, adapté à un objet déformable. La ligne 4 calcule uniquement la jacobienne de la forme actuelle. On ne calcule pas les jacobiennes des formes futures - les expériences conduites en simulation n'ont pas convergé lorsque nous estimions les jacobiennes des formes futures. Ce résultat est contre-intuitif, car à première vue, cela aurait dû au contraire améliorer la stratégie d'asservissement de forme.

La ligne 6 définit l'horizon restant comme étant le minimum entre la valeur de la fenêtre glissante et le nombre restant d'itérations jusqu'à l'horizon temporel. Cette fenêtre glissante, liée à l'horizon temporel, est décrite plus en détail dans la section 3.4.1.

La ligne 8 lance la boucle récursive sur la fenêtre glissante pour le calcul de S. Ainsi, S est calculé sur une fenêtre plus courte plutôt que sur l'horizon temporel complet. Cette boucle récursive est la cause de la charge en calcul générée par la commande. La jacobienne est constante pendant la boucle récursive. Cela permet de réduire le coût en calcul, en plus d'être une limite venant du résultat contre-intuitif exposé précédemment : des hypothétiques approches de modèles permettant d'obtenir une Jacobienne dépassant cette limitation n'améliorerait donc pas trivialement la

stratégie d'asservissement de forme. La boucle récursive est également écrite en tenant compte d'éventuelles matrices Q et R variables dans le temps.

Algorithme 1 Stratégie `OSS`

Input :

x_d ▷ Desired shape
dt ▷ Time step (seconds)
t_h ▷ Time horizon (number of iterations)
sw ▷ Sliding window (number of iterations)
Q ▷ Shape constraints (matrix or matrices)
R ▷ Control constraints (matrix or matrices)

for k=1 :t_h **do** ▷ control loop over time horizon
  x = `PerceiveShape`(object)
  $e = x - x_d$ ▷ computes the shape error
  J = `ComputeDeformationJacobian`(x)
  $G = dt\,J$
  r_h = `min`(sw, t_h - k) ▷ remaining horizon
  $S = Q(k + r_h)$ ▷ initialize S for the recursive loop
  for i=1 :r_h **do** ▷ recursive loop over sliding window
    $R_i = R(k + r_h - \mathrm{i} + 1)$ ▷ Set current R
    $Q_i = Q(k + r_h - \mathrm{i})$ ▷ Set current Q
    $S = S - (SG(G^\top SG + R_i)^{-1})G^\top S + Q_i$ ▷ Recursive computation of S
  end for
  $K = (G^\top SG + R)^{-1}G^\top S$
  $u = -K\,e$ ▷ control law
  `MoveRobots`(u)
end for

3.4 Expériences et Résultats

Nous avons validé la stratégie `OSS` avec une tâche simulée et trois tâches réelles. La tâche simulée est l'asservissement de forme d'un maillage surfacique simplement translaté, correspondant au comportement de la gomme manipulée lors de la première tâche réelle. Elle est utilisée pour l'analyse du temps de convergence. Les tâches réelles sont :

— L'assemblage d'une bande de caoutchouc. Dans cette tâche, nous plaçons une pièce de caoutchouc en prolongement d'une autre pièce fixée sur un cylindre. Cette tâche est détaillée dans le chapitre 1. La commande `OSS` a été spécifiquement pensée pour accomplir cette tâche.
— La mise en boîte d'un poster. Dans cette tâche, le poster doit être déformé avant d'être mis dans la boîte.
— L'apprentissage par démonstration des paramètres de la tâche.

Dans chaque tâche, nous comparons la stratégie `OSS` avec ARAP-SS [SB+22].

3.4.1 Tâche simulée

Dans cette tâche, nous souhaitons donner la priorité à la convergence d'un côté par rapport à l'autre, ainsi qu'au centre par rapport aux côtés. La pondération, représentée par une carte thermique dans la figure 3.2 illustre les gains de notre matrice Q sur les nœuds du maillage. Cette

matrice encode le comportement de déformation de la forme tel qu'illustré dans la figure 3.3. Nous nous concentrerons dans cette section sur l'effet des paramètres de l'horizon temporel et de la fenêtre glissante.

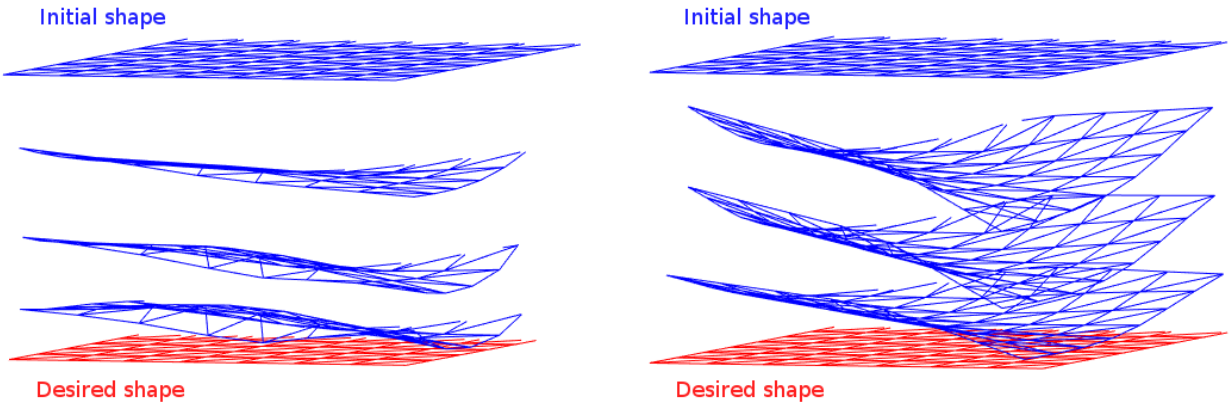

FIGURE 3.3: Convergence de forme ARAP-SS et OSS au cours de l'exécution

ARAP-SS (gauche) et OSS (droite). La stratégie OSS utilise un comportement de déformation encodé.

Configuration de la simulation

Nous avons utilisé un maillage de dimensions 8×8. Elle est déformée par deux préhenseurs qui saisissent les côtés et se déplacent librement. La fréquence de contrôle est fixée à $10Hz$. Ainsi, le pas de temps dt est de $0.1s$. Par conséquent, un horizon temporel avec 100 itérations correspondrait à 10 secondes. Nous avons défini la forme désirée comme un maillage plan. La forme initiale est le même maillage plan déplacée de $30cm$ vers le haut. Nous avons utilisé ARAP pour modéliser l'objet et calculer le Jacobien. La matrice R est définie comme étant égale à $0,001$ fois une matrice identité. Ce choix permet une liberté maximale pour le contrôle - avec ce paramétrage, le comportement de l'état importe ; celui des effecteurs n'importe presque pas.

Horizon temporel pour le temps de convergence de la tâche

Nous faisons varier le paramètre de l'horizon temporel de $10s$ à $40s$ afin d'analyser les temps de convergence de la tâche. Nous avons fixé les tailles de la fenêtre glissante égales aux horizons temporels (*e.g.*, $t_h = sw = 10s$, $t_h = sw = 20s$, et ainsi de suite). Les résultats de la simulation sont présentés Fig. 3.4. On peut observer que les temps de convergence de la tâche imposés par les horizons temporels sont presque respectés.

Impact de la taille de la fenêtre glissante

Nous rappelons que plus la taille de la fenêtre glissante est petite, plus rapide est le calcul de la loi de commande. Par conséquent, nous faisons varier les tailles des fenêtres glissantes de $t_h/2$ à $t_h/6$ afin d'analyser la relation entre la taille de la fenêtre glissante et le temps de convergence de la tâche. Nous chercherons le meilleur compromis entre le coût de calcul de la loi de commande et l'assurance du temps de convergence de la tâche. La figure 3.5 montre les résultats simulés pour mettre en évidence la relation entre les tailles de fenêtres glissantes sur les différents horizons temporels et les comportements de convergence, comme illustré Fig. 3.4. Un rapport de $t_h/4$ permet

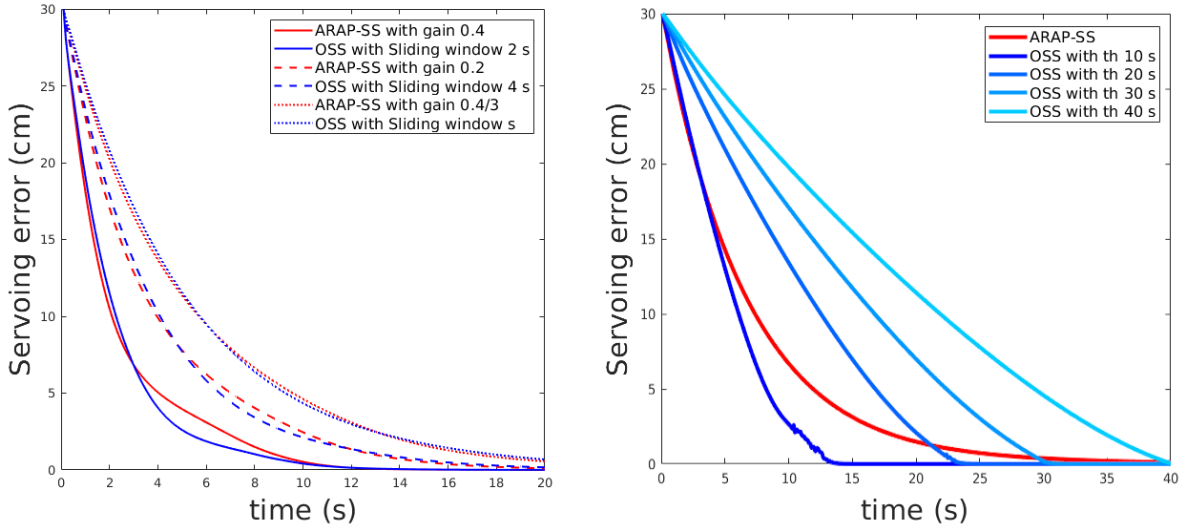

FIGURE 3.4: Erreur d'asservissement en fonction du temps

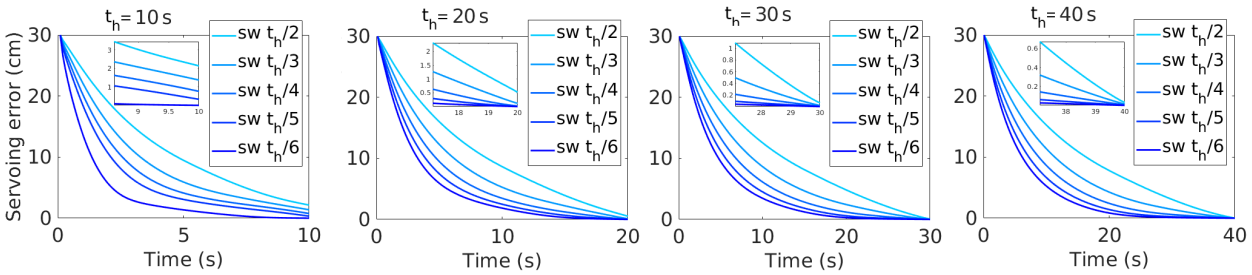

FIGURE 3.5: OSS scheme servoing error

OSS scheme servoing error versus time for the analysis of task convergence times. Small windows zoom the last few seconds.

d'obtenir un bon compromis entre le coût de calcul et l'assurance du temps de convergence de la tâche.

Dans la pratique, il faut dimensionner la fenêtre glissante pour que le coût en calcul de la commande soit légèrement inférieur au coût en calcul de la reconstruction de forme par les algorithmes de perception, qui bornent le temps disponible pour calculer les trajectoires en vitesse des robots.

3.4.2 Tâche d'assemblage de bandes de gomme

Cette tâche est une approximation du cas d'usage réel, présenté dans le chapitre 1. Le problème est simplifié pour se focaliser sur les questions de contrôle, afin de valider la faisabilité de l'application de la stratégie OSS au cas d'usage.

3.4.2.1 Cas d'utilisation industriel

La bande de caoutchouc est enroulée autour d'un support cylindrique. L'opérateur humain aligne une extrémité mobile sur une extrémité fixe, comme illustré à la figure 3.6. Il suit deux étapes : il commence par déposer l'arrière de la bande sur le rouleau, puis il dépose l'avant de la bande en commençant par le milieu tout en étalant progressivement les bords latéraux pour les faire correspondre au bord fixe. Notre objectif est d'automatiser cette tâche manuelle.

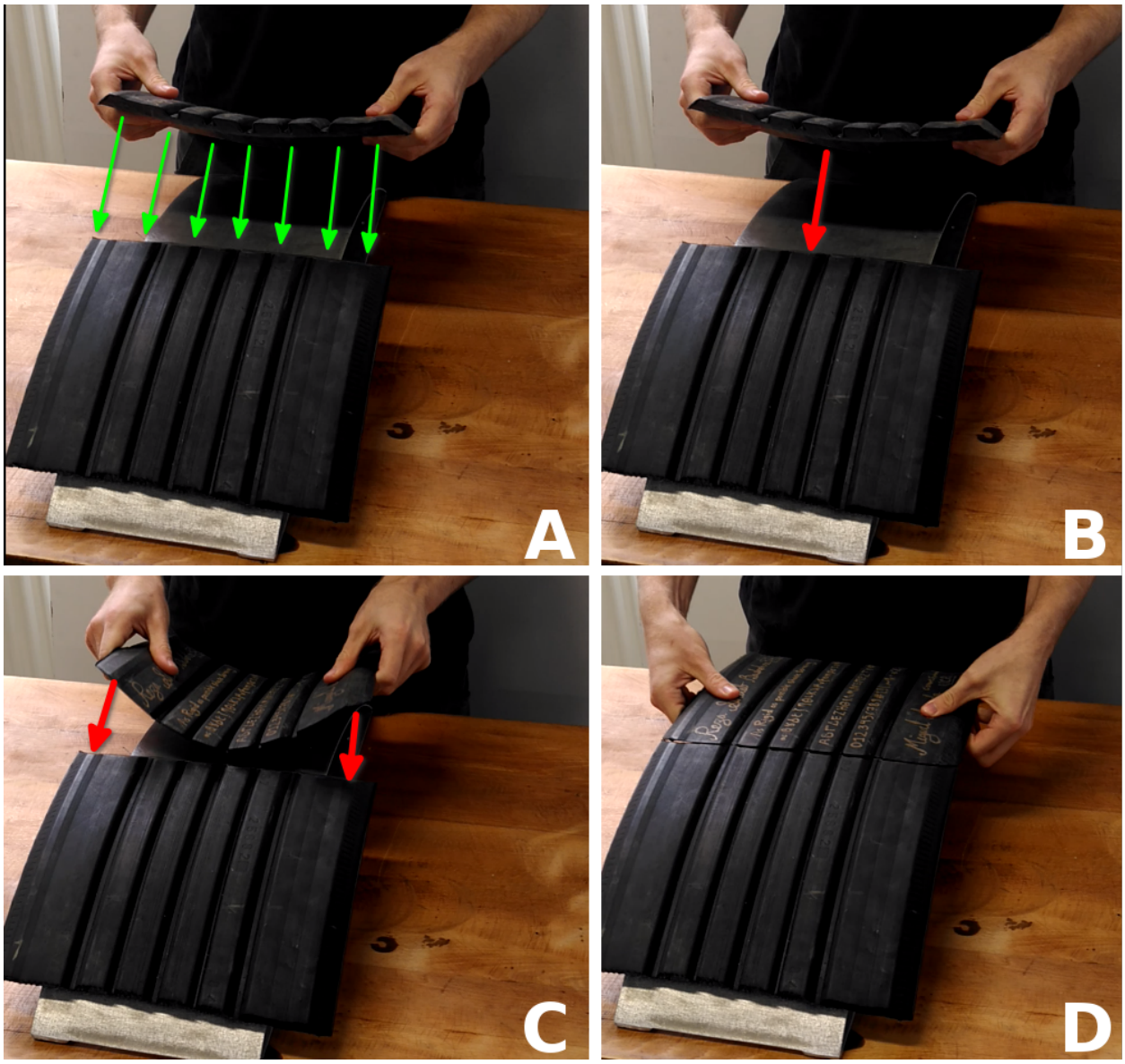

FIGURE 3.6: Assemblage manuel de la gomme comme fait en usine.

Afin d'obtenir la forme désirée, telle que présentée à la figure 3.7, nous avons d'abord aligné manuellement la pièce mobile de caoutchouc avec la pièce fixe sur un cylindre, puis suivi la forme avec l'algorithme SfT. Bien que la caméra soit RGB-D, SfT n'utilise que le canal RGB.

3.4.2.2 Méthodologie expérimentale

Nous avons analysé le comportement des stratégies d'asservissement de forme pour différentes configurations initiales. Nous avons testé 3 scénarios. Dans le premier scénario, la configuration initiale est une forme plate déplacée par rapport à la forme désirée. Dans le deuxième scénario, la configuration initiale est une forme convexe déplacée. Dans le troisième scénario, la configuration initiale est une forme concave déplacée.

Pour ce faire, nous avons tout d'abord créé une matrice Q dont la représentation sous forme de carte de chaleur est très similaire à celle présentée à la figure 3.2 – associée à un maillage de 7x10 nœuds et non pas 8x8 – afin de pouvoir reproduire les gestes de l'opérateur humain. Dans le but de quantifier l'asservissement de forme, nous avons ensuite créé des trajectoires de référence pour 9 nœuds. Ces nœuds sont numérotés de 0, 3, 6, 35, 38, 41, 63, 66 et 69 sur la figure 3.7.

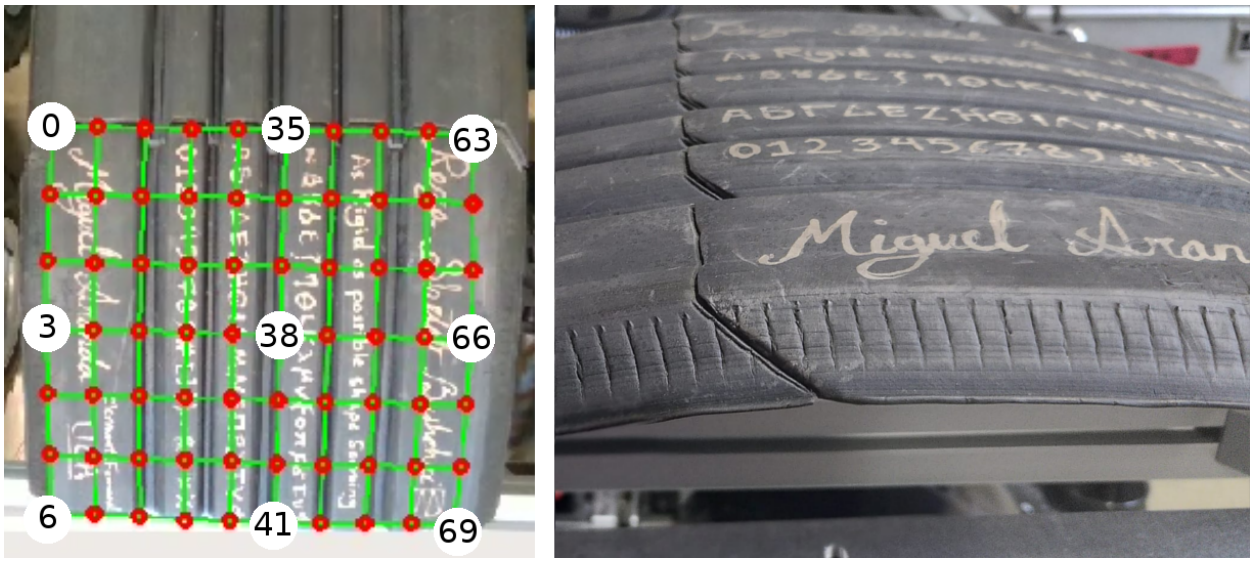

FIGURE 3.7: Maillage perçu de la bande de roulement

Gauche : configuration souhaitée. Droite : vue latérale de la même configuration. Une vidéo multiplexée montrant la tâche effectuée sous plusieurs angles, ainsi que le maillage, est disponible sur `https://youtu.be/yDiV5dDr5To`

Leur trajectoires sont observées par SfT tout en effectuant manuellement la tâche à partir d'une configuration initiale similaire au premier scénario. Les trajectoires observées des 9 nœuds sont ensuite utilisées comme références implicites pour le comportement de convergence dans les trois scénarios, indépendamment des formes initiales différentes.

3.4.2.3 Résultats

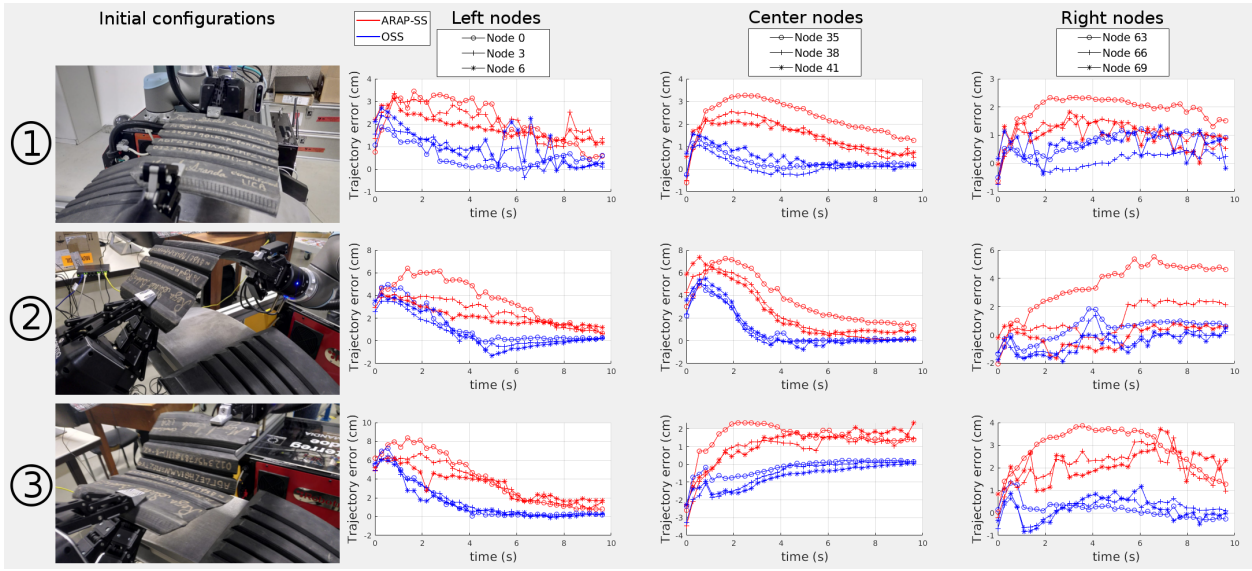

FIGURE 3.8: Comparaison des erreurs de trajectoire

Erreurs de trajectoires des nœuds dans 3 scénarios avec $t_h = 10s$, $sw = 2s$. *Rouge : Erreur avec stratégie ARAP-SS. Bleu : Erreur avec stratégie* `OSS`. *Une vidéo de l'expérience comparant les deux stratégies est disponible sur* `https://www.youtube.com/watch?v=7wFasZ7imEY`.

Nous avons tracé les résultats des 3 scénarios dans la figure 3.8. Chaque ligne présente les résultats d'un scénario. La première colonne montre les configurations initiales. La deuxième colonne quantifie dans quelle mesure les nœuds 0, 3 et 6 ont suivi leurs comportements de convergence de référence respectifs. Les courbes représentent la différence entre les trajectoires des nœuds et leurs trajectoires de référence, comme expliqué précédemment. De même, les deuxième et troisième colonnes quantifient dans quelle mesure les nœuds 35, 38, 41 et 63, 66, 69 ont suivi leurs comportements attendus.

Les courbes représentent la différence entre les trajectoires réelles et désirées des nœuds. Plus les points sont proches du zéro, plus le comportement de référence est respecté. La stratégie proposée surpasse la stratégie proportionnelle dans tous les scénarios. Même si quelques pointes sont présentes sur les courbes dues aux bruits de perception, la forme de l'objet a été asservie de manière stable vers la forme cible.

Un résultat inattendu a été mis en avant lors des expériences : l'intégration des poids de la matrice R permet aux effecteurs d'avoir un comportement plus proche de celui qu'aurait un opérateur humain accomplissant la tâche : en effectuant la manipulation en se concentrant sur les directions privilégiées ou non, on peut pondérer les DDL représentés par cette matrices. Ceci permet de réduire l'espace de travail nécessaire au robot, et de réduire le risque de collisions.

3.4.3 Mise en boîte d'un poster

Lors de nos tests en simulation présenté dans la section 3.4.1, nous avons constaté une faiblesse dans la stratégie proportionnelle d'asservissement de forme : le système impose une déformation sur une simple translation, comme le montre la figure 3.9.

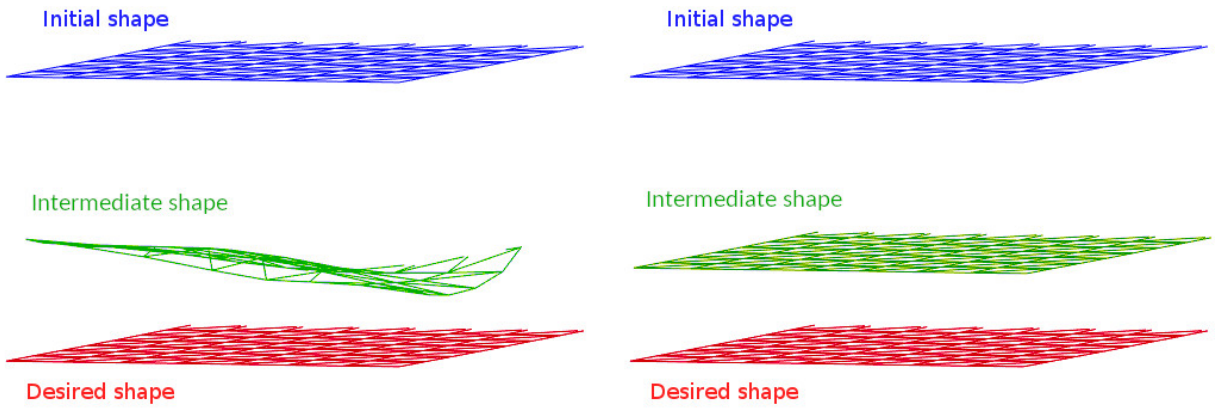

FIGURE 3.9: Biais imposant une déformation à une tâche de translation

Gauche : contrôleur ARAP-proportionnel. Droite : même tâche, mais le contrôleur impose uniquement une translation à l'objet

Cette faiblesse provient de biais dans l'implémentation du modèle ARAP ainsi que des problèmes d'égalités imparfaites de nombres flottant. Bien sûr, il se trouve amplifié dans le cas d'utilisation réel dû aux bruits de mesure. Ce constat intriguant a mené à une nouvelle technique : découpler l'erreur de forme de l'erreur de position.

3.4.3.1 Description de la tâche

Dans cette tâche, nous essayons de mettre un morceau de papier dans une boîte trop petite pour cela. La configuration, illustrée à la figure 3.10, est composée de deux robots UR10 tenant

une feuille de papier au format A4 affichant une texture, d'un capteur visuel et de la boîte elle-même. La texture utilisée est traditionnellement utilisée dans les tâches de perception, il s'agit de la couverture du premier numéro de la bande dessinée emblématique "The Amazing Spider-Man". Notre capteur pour cette tâche est une caméra Intel Realsense.

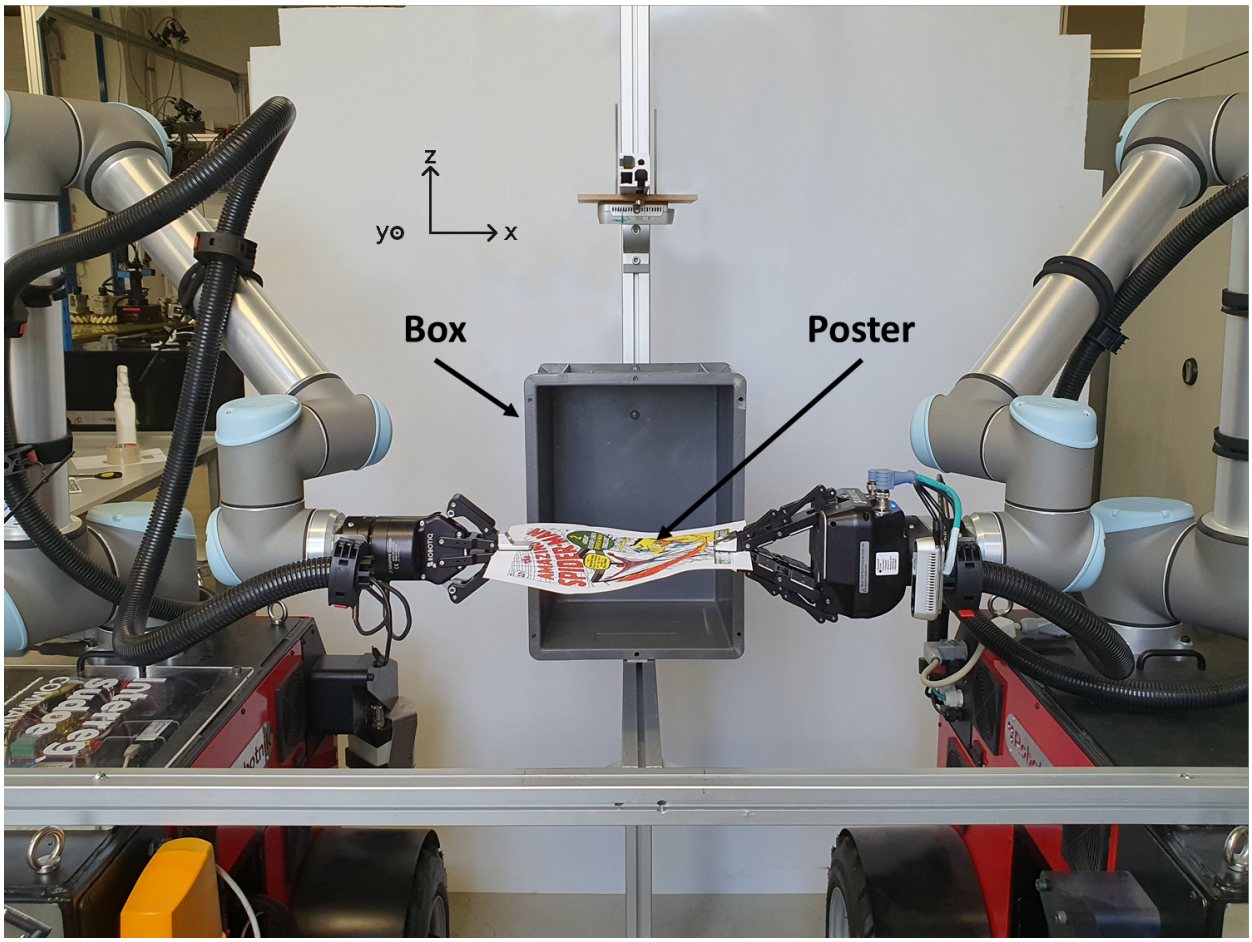

FIGURE 3.10: Configuration de la tâche de mise en boîte d'un poster

Nous définissons le succès de la tâche par la mise en boîte du poster sans toucher les bords. Cette tâche est difficile car notre forme cible ne permet pas beaucoup de marge, comme illustré à la figure 3.11. Cette tâche a été conçue pour mettre en évidence les points forts d'`OSS` par rapport au contrôle proportionnel traditionnel.

Pour cette tâche, le problème a été formulé tel que l'erreur de position est directement corrélée à l'axe y, alors que l'erreur de forme est corrélée aux axes x et z.

Au cours de nos expérimentation, nous avons déterminé une mètrique permettant de quantifier numériquement le succès ou l'échec de la tâche, que nous montrons dans la Fig. 3.13. Les métriques représentées sont l'erreur moyenne de forme des nœuds (x error) en fonction de l'erreur de position de l'objet (y error). La tâche est difficile : l'objet rentre dans la boîte quand l'erreur de position est de $y = 4\,cm$. A cet instant critique, l'erreur de forme doit être faible – $1,25\,cm$.

3.4.3.2 Expériences

Pour l'application de cette stratégie, le contrôleur proportionnel échoue, comme le montre la figure 3.12. Si nous définissons la matrice Q comme étant l'identité, le comportement de la stratégie `OSS` est similaire et la tâche échoue également. Nous présenterons deux stratégies parvenant à résoudre la tâche.

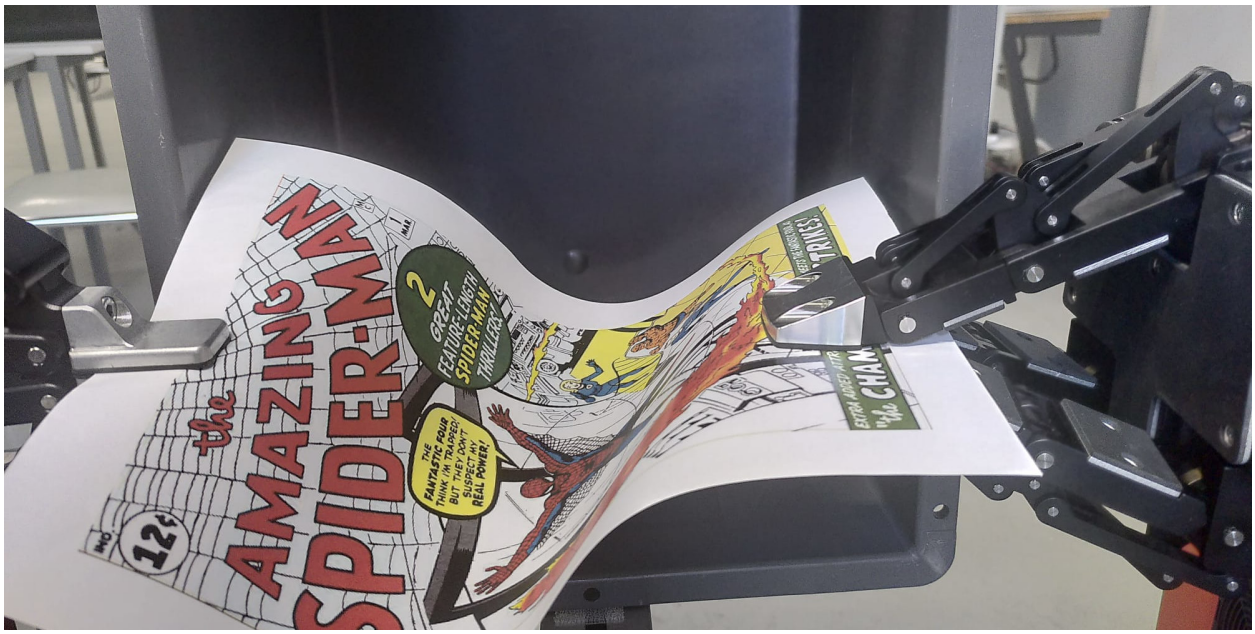

FIGURE 3.11: Forme cible pour la mise en boîte

FIGURE 3.12: Vue du dessus de la tâche à l'instant critique

L'instant critique est défini comme le moment où le poster commence à entrer dans la boîte. Gauche : contrôleur ARAP-Proportionnel. Droite : contrôleur Optimal.

Stratégie `OSS` en deux phases Dans cette stratégie, nous utilisons pour la première moitié de la tâche la matrice Q décrite dans l'Algorithme 2 avec un gain $k = 10000$, la seconde moitié avec la matrice Q égale à la matrice identité.

L'intérêt de cette valeur de k est de pondérer 100 fois plus l'erreur de forme que l'erreur de position - cela permet d'asservir la forme de l'objet pendant la première moitié de la tâche, puis d'effectuer une translation vers la forme cible pendant la seconde moitié de la tâche.

Stratégie `OSS` lisse Pour l'application de cette stratégie, nous utilisons la matrice Q décrite dans l'Algorithme 2 avec un gain $k = 9$. Cette valeur de k a été déterminée empiriquement. Cette pondération permet d'accomplir la tâche de manière fiable et répétable au plus près des bords de la boîte, comme montré dans la Fig. 3.12.

Algorithme 2 Création automatique de la matrice Q

Input :

k ▷ Shape error gain
$mesh\_x$ ▷ Number of nodes on the x axis
$mesh\_y$ ▷ Number of nodes on the y axis

Q = Zero(3*$mesh\_x$*$mesh\_y$) ▷ Initialize Q as Zero matrix
for $node\_y\_index$=1 :$mesh\_y$ **do**
  for $node\_x\_index$=1 :$mesh_x$ **do**
    $n = mesh_x$ * $node\_y\_index + node\_x\_index$;
    $Q(n,n) = \mathrm{k}$;
    $Q(n+1,n+1) = 1$;
    $Q(n+2,n+2) = \mathrm{k}$;
  end for
end for

3.4.3.3 Résultats

La Fig. 3.13 montre les résultats de cette expérience, comparant la stratégie Proportionnelle à OSS avec les deux matrices Q proposées ci-dessus. Le seuil critique est représenté par la ligne en pointillés noirs sur la figure. Ce graphique met en avant les résultats expérimentaux constatés :

— Le contrôleur proportionnel (en rouge) est au-dessus du seuil à l'instant critique, ce qui se traduit dans le monde réel par une collision à l'entrée dans la boîte.
— Le contrôleur `OSS` en deux phases (en jaune) effectue l'asservissement de forme avant celui de position - permettant la meilleure marge possible pour effectuer la tâche.
— Le contrôleur `OSS` lisse (en bleu) est 4 *mm* au-dessous du seuil à l'instant critique, ce qui représente une marge de 2 *mm* de chaque côté de la feuille.

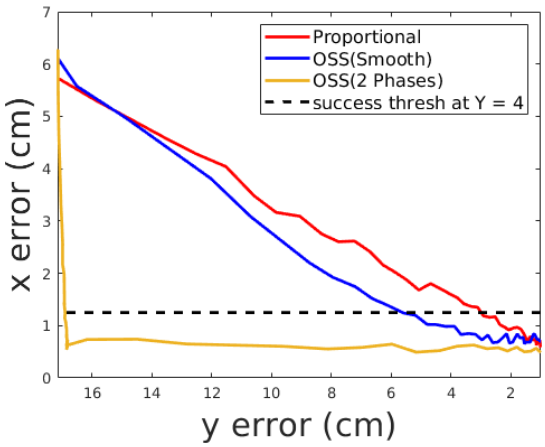

FIGURE 3.13: Erreur de forme en fonction de l'erreur de position

Le découpage intuitif et paramétrable du problème en "erreur de forme" et "erreur de position" est généralisable à d'autres stratégies de manipulation d'objets déformables, comme par exemple

le contrôleur proportionnel adapté à la tâche présenté section 4.5.4, où la forme de la section manipulée est imposée par un mouvement en rotation pendant que la tâche est effectuée avec des translations. Cependant, pour effectuer cette séparation en réglant manuellement les poids des matrices, il faut soit définir le problème d'une manière à le rendre directement compatible, en corrélant complètement une coordonnée à l'erreur de position par exemple, soit effectuer des transformations de repère de l'intégralité du maillage, appliquer la commande, et effectuer la transformation inverse pour les ordres à appliquer au robot. D'autres possibilités existent : la question de trouver une transformation à appliquer aux matrices Q et R pour intégrer implicitement des poids directionnels reste ouverte. Nous avons choisi de concentrer nos efforts sur une autre piste, permettant d'éliminer le besoin du paramétrage manuel.

3.4.4 Apprentissage par démonstration des paramètres des matrices de poids avec un algorithme génétique

Comme dit précédemment, le principal inconvénient de la stratégie `OSS` vient du paramétrage nécessaire des matrices de poids. Nous proposons donc dans cette section une méthodologie permettant d'effectuer un apprentissage par démonstration de ces paramètres.

3.4.4.1 Description de la tâche

Cette tâche est une simulation de manipulation de poutre déformable. Notre poutre simulée mesure 1m de long, et commence en position horizontale. L'extrémité gauche de la poutre est encastrée - impossible pour les nœuds 1 et 2 de bouger, l'extrémité droite manipulée : le préhenseur simulé a saisi l'objet entre les nœuds 9 et 10. Le préhenseur simulé peut effectuer des translations suivant les axes x et y, ainsi que des rotations dans le plan. Les rotations impactent donc la position de ces deux nœuds. Le modèle est appliqué de manière itérative sur les triplets de nœuds. Nous avons généré une trajectoire de déformation de la poutre, avec les commandes générées par l'algorithme 3, résultant en la trajectoire de déformation montrée Fig. 3.14. Les stratégies de commandes ont uniquement accès à la forme finale désirée dans la boucle de contrôle. Le but est d'obtenir une commande capable de suivre la trajectoire de déformation. Cette trajectoire de déformation à suivre est disponible pendant les phases off-line de la synthèse du contrôleur, mais ne l'est plus dans les phases on-line.

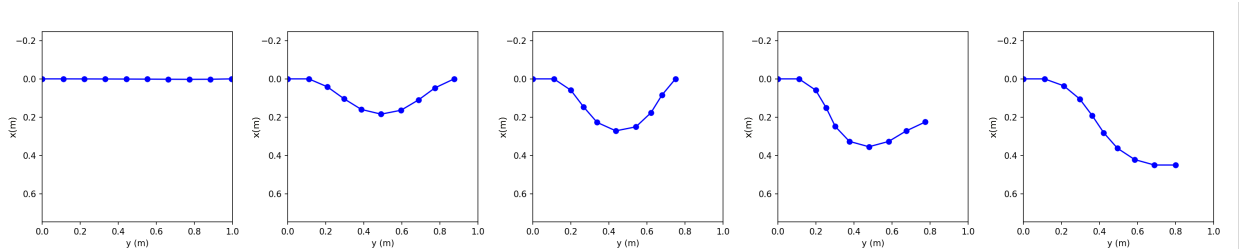

FIGURE 3.14: Déformation cible aux instants 1, 25, 50, 75 et 100

3.4.4.2 Expériences

Nous avons simulé le comportement de notre objet déformable avec la formulation linéaire du modèle ARAP, en 2 dimensions. Le choix du modèle et du nombre de nœuds du maillage a été effectué dans une optique de simplification : en commandant le comportement des 8 nœuds libres avec un préhenseur saisissant les deux derniers et capable de translation en x et y ainsi que

Algorithme 3 Génération de la forme cible

```
1:  for t=1 :100 do
2:      if t < 50 then
3:          cmd.vx = 0;
4:          cmd.vy = -0.005;
5:          cmd.vθ = π*1/180;
6:      else
7:          cmd.vx = 0.009;
8:          cmd.vy = 0.001;
9:          cmd.vθ = -π*1/180;
10:     end if
11:     deformRope(cmd);
12: end for
```

de rotation, la matrice Q devient de taille $(16,16)$; la matrice R devient de taille $(3,3)$; la matrice J devient de taille $(16,3)$. L'objectif assumé est de réduire la complexité calculatoire pour la phase exploratoire et de mise au point, permettant des simulations bien plus rapides pour permettre l'élaboration d'une stratégie fonctionnelle.

Pour effectuer l'apprentissage par démonstration, nous proposons un algorithme génétique comme présenté dans l'algorithme 4. Cet algorithme propose une implémentation possible de l'apprentissage par déformation. Le concept est de définir les matrices Q et R comme un génome, que nous soumettrons à une forme de sélection naturelle : les plus adaptés à notre fonction de coût se reproduiront, les autres disparaîtront. Tout d'abord, nous simulons le résultat de la commande `OSS` issue des matrices Identités, et conservons ce score. Ensuite, nous faisons muter les génomes, avec les fonctions de mutation définies dans l'algorithme 5. Le génome est simulé sur l'horizon de temps de la simulation avec `OSS`, permettant d'obtenir le score associé au génome. Nous concatenons notre génome de la génération précédente à celui de la génération actuelle, et nous le trions. Nous ne garderons que ceux présentant l'erreur la plus faible (ligne 8), que nous soumettons à nouveau à des mutations (lignes 16 et 18).

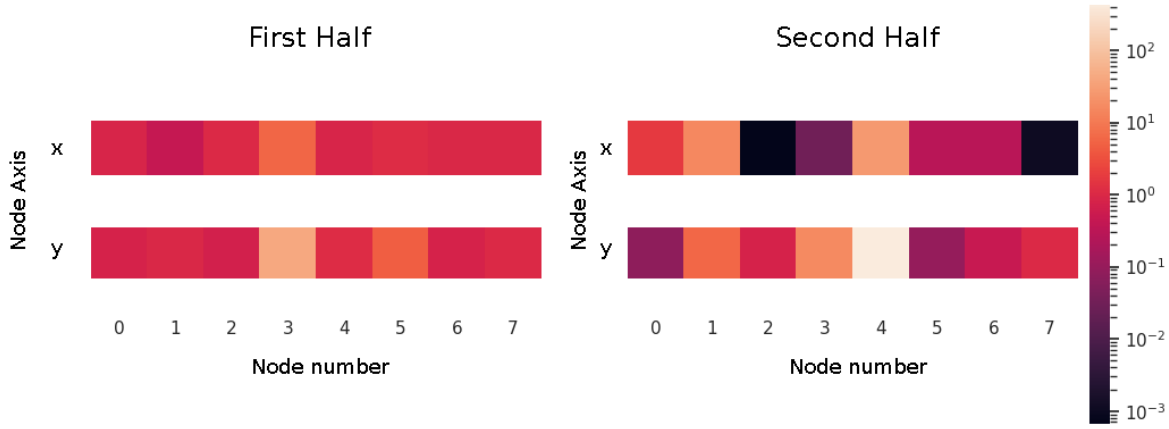

FIGURE 3.15: Valeurs indentifiées des coefficients de la matrice Q pour l'objet linéique

L'identification des paramètres par algorithme génétique donne des coefficients décorrélés des axes.

Algorithme 4 Algorithme génétique proposé

```
    main(QBegi, RBegi)
 1: QFin = Matrix.ID(16)
 2: RFin = Matrix.ID(3)
 3: trajError = oss_noshow(rope, QBegi, RBegi, QFin, RFin)
 4: for index = 0 :4 do
 5:     geneList.append(QFin, RFin, trajError)
 6: end for
 7: for mutation = 1 :20 do
 8:     geneList.sortByError()
 9:     geneList = geneList[ :5]
10:     newgeneList = []
11:     for index = 0 :10 do
12:         QFin= geneList[index%5].Q)
13:         RFin= geneList[index%5].R)
14:         for index = 1 :10 do
15:             if (mutation <= 10) then
16:                 mutateBeaucoup(QFin, RFin)
17:             else
18:                 mutate(QFin, RFin)
19:             end if
20:             trajError = oss_noshow(rope, QBegi, RBegi, QFin, RFin)
21:             geneList.append(QFin, RFin, trajError)
22:         end for
23:     end for
24:     geneList = geneList+newgeneList
25: end for
26: geneList.sortByError()
27: oss_show(rope, QBegi, RBegi, geneList[0].Q, geneList[0].R)
```

La stratégie de mutation est en deux phases : durant la première, un fort taux de mutation avec un fort impact permet une bonne exploration de l'espace des paramètres. Durant la seconde, le taux plus faible avec des bornes biens plus resserrées autour des paramètres actuels a un rôle de recherche de réglages fins. Nous avons créé deux fonctions différentes au comportement similaire pour une meilleure lisibilité du fonctionnement de l'algorithme 4.

Cette stratégie a été appliquée deux fois, pour obtenir deux set de matrices Q et R , chacun appliqués à une moitié de la tâche. Les matrices Q résultantes sont montrées dans la Fig. 3.15. Le nombre de modalités du système et leur durée d'application sont pour l'instant des métaparamètre, qu'il serait envisageable de faire rentrer dans le génome.

Algorithme 5 Fonctions de mutation

mutateBeaucoup(Q,R) :

for i=0 :15 **do**
    chance = `random.int`(1,100)
    if (*chance* <= 50) **then**
        mut = `random.uniform`(-2,2)
        $Q[i,i] = Q[i,i] * 10\hat{}(mut)$
    end if
end for
for i=0 :2 **do**
    chance = `random.int`(1,100)
    if (*chance* <= 40) **then**
        mut = `random.uniform`(-2,2)
        $R[i,i] = R[i,i] * 10\hat{}(mut)$
    end if
end for

mutate(Q,R) :

for i=0 :15 **do**
    chance = `random.int`(1,100)
    if (*chance* <= 10) **then**
        mut = `random.uniform`(0.8,1.2)
        $Q[i,i] = Q[i,i]$*mut
    end if
end for
for i=0 :2 **do**
    chance = `random.int`(1,100)
    if (*chance* <= 20) **then**
        mut = `random.uniform`(0.8,1.2)
        $R[i,i] = R[i,i]$*mut
    end if
end for

opti_noshow(rope, Q, R, QBegi, RBegi) :

$trajError$ = [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]
$Q\_OC = QBegi$
$R\_OC = RBegi$
for t=1 :100 **do**
    if (t == 50) **then**
        $Q\_OC = Q$
        $R\_OC = R$
    end if
    $trajError$ += `abs`(rope.currentShape - targetTrajectory(t)
    $j = rope$.`getJaco()`
    cmd = `OSS`($rope.targetShape, 0.1, 100-t, 100, Q\_OC, R\_OC, j, rope.currentShape$)
    $rope$.`deformARAP`(cmd)
end for
return $trajError$

3.4.4.3 Résultats

La stratégie proposée permet de suivre correctement la trajectoire de déformation, comme montré dans la Fig. 3.16. La trajectoire de déformation elle-même a été conçue pour être difficile à apprendre : la trajectoire est constituée d'échelons, que la commande ne peut supposément pas effectuer avec la formulation proposée. Les commandes de référence, ainsi que celles générées par nos contrôleurs sont représentées Fig. 3.17.

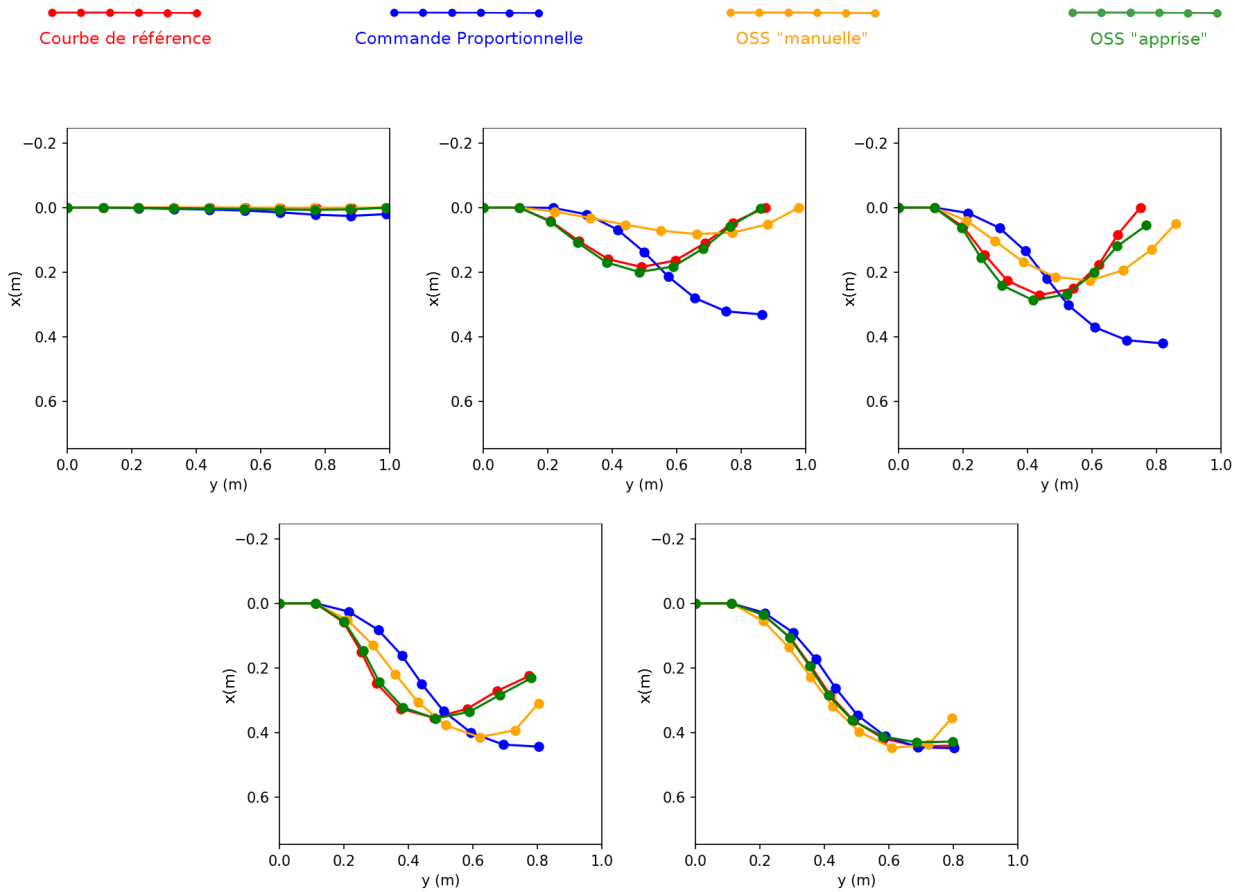

FIGURE 3.16: Déformation aux instants 1, 25, 50, 75 et 100

Rouge : forme cible. Bleu : contrôleur proportionnel. Orange : `OSS` avec matrices réglées à la main. Vert : `OSS` avec matrices apprises par démonstration

Cette tâche nous a permis de constater un cas où le contrôleur proportionnel n'arrive pas à accomplir une forme que nous savons atteignable : en effet, nos premiers développements ont résulté en un contrôleur conservant une erreur statique de position en fin de tâche, comme pour les matrices générant la forme orange dans la Fig. 3.16. Nous avons donc analysé cette partie de la tâche plus en détail, comme montré dans la Fig. 3.18. Nous avons essayé plusieurs stratégies, en particulier d'imposer des matrices Identités pour finir la tâche, ce qui n'a pas fonctionné. La solution finale que nous proposons est inspirée par nos autres travaux de contrôle de forme (voir Section 4.5.4, en particulier le contrôleur proportionnel adapté à la tâche). Dans ce cas, la forme atteinte est très proche de la forme désirée. Nous considérons que la tâche est solvable uniquement en contrôlant l'erreur d'angle et de position des deux derniers nœuds. Cette modification permet de s'assurer de converger vers la forme désirée.

Pour valider l'apprentissage, nous avons également ajouté un bruit gaussien blanc à la "perception" avant d'appliquer le contrôle. L'amplitude du bruit est de -2cm à +2cm d'erreur - de l'ordre de 20 fois ce que nous pouvons raisonnablement obtenir avec une perception adaptée, comme celles testées dans la section 4.6. Les commandes bruitées sont montrées Fig. 3.19. La commande résultant des matrices apprises semble être celle souffrant le plus de ces bruits de perception - en particulier pour les ordres de commande angulaire du robot. Cependant, les performances de la commande avec matrices apprises par démonstration restent largement supérieures à celles des autres commandes, ce que nous montrons dans la Fig. 3.20. Meme si le bruit dégrade la qualité du suivi de trajectoire lors de la deuxième moitié de la commande, la performance du suivi de trajectoire de la stratégie proposée reste inégalée.

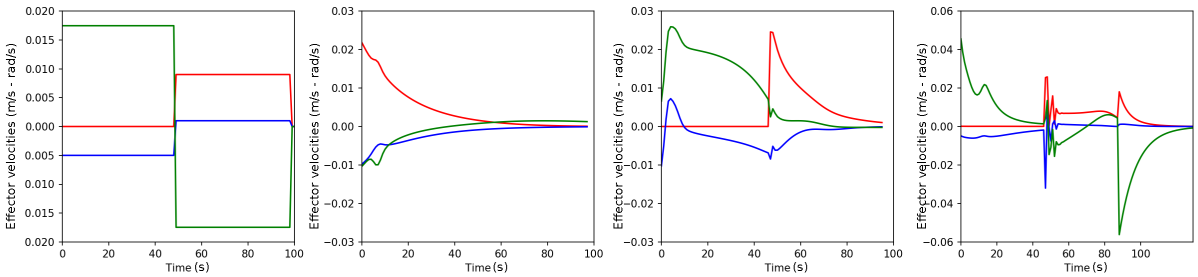

FIGURE 3.17: Commandes générées par stratégie d'asservissement de forme

Rouge : Vitesse en X. Bleu : Vitesse en Y. Vert : Vitesse angulaire.

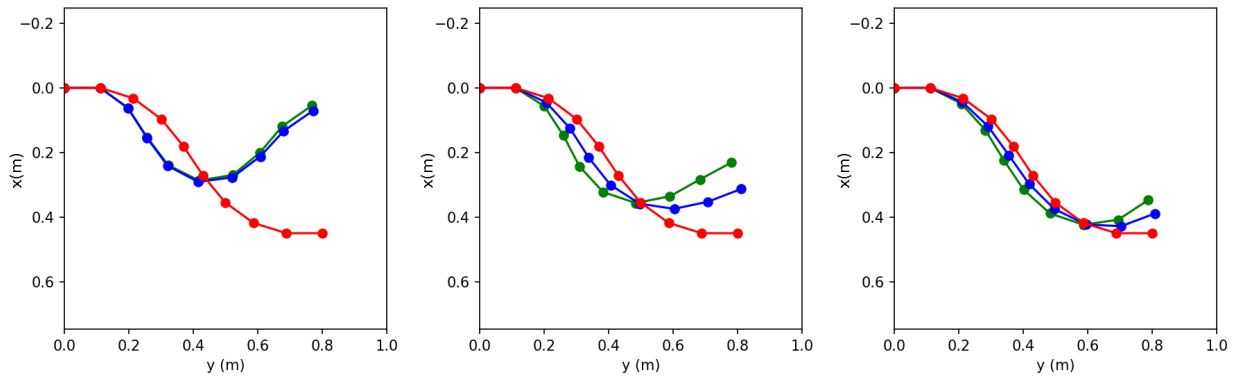

FIGURE 3.18: Echec de convergence des contrôleurs, instants 51, 75 et 100

Rouge : forme cible. Bleu : contrôleur proportionnel. Orange : `OSS` avec matrices réglées à la main.
Le contrôleur ARAP-Proportionnel ne parvient pas à converger depuis la forme obtenue après la première moitié de la tâche. Le contrôleur optimal non plus - son erreur relative est même amplifiée.

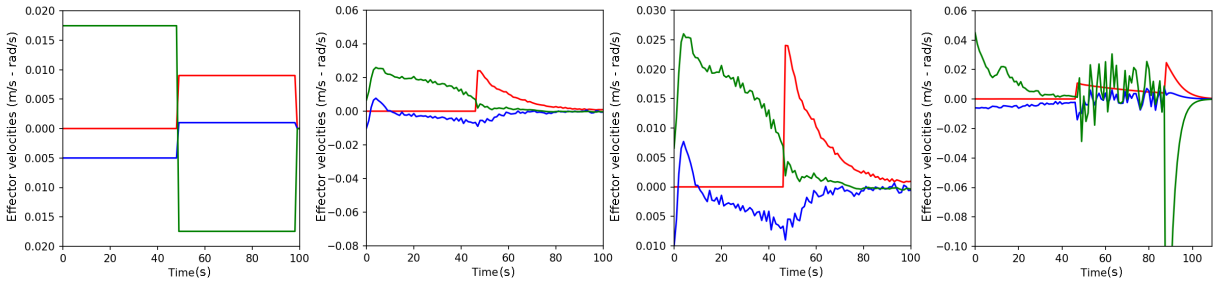

FIGURE 3.19: Commandes générées avec un bruit blanc gaussien de perception

Rouge : Vitesse en X. Bleu : Vitesse en Y. Vert : Vitesse angulaire.

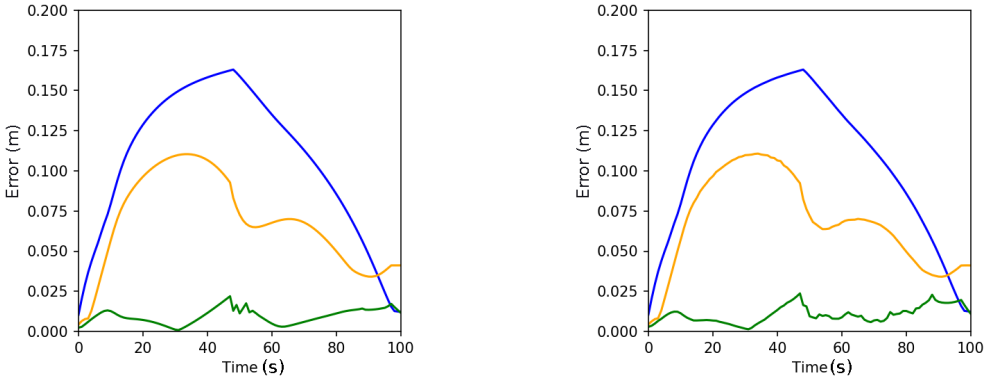

FIGURE 3.20: Erreur de trajectoire des différentes stratégies de contrôle

Gauche : erreur en simulation. Droite : ajout d'un bruit blanc.
Bleu : contrôleur proportionnel. Orange : `OSS` avec matrices réglées à la main. Vert : `OSS` avec matrices apprises par démonstration.
Erreur définie comme la distance de chaque nœud avec la position du nœud de référence, divisés par le nombre de nœuds

3.5 Conclusion

Dans ce chapitre, nous avons présenté une stratégie de contrôle optimal capable d'effectuer un asservissement de forme d'objet déformable en prenant en compte la trajectoire prise par les formes de l'objet pour atteindre l'état final désiré. Ces travaux - tout du moins les sections théoriques et la tâche d'assemblage de gomme - ont été présentés à la conférence IROS 2022 [Gir+22].

— La stratégie permet d'effectuer des tâches complexes d'asservissement de forme, en remplissant sa promesse de suivi de trajectoire de déformation.
— Les mouvements générés par les robots peuvent suivre des contraintes pour s'adapter à un espace de travail plus restreint.
— La stratégie permet de décorréler l'erreur de forme à l'erreur de position en choisissant des repères de travail adapté.
— Il est possible d'effectuer un apprentissage par démonstration des paramètres de la stratégie, permettant d'allier les forces d'apprentissage par démonstration à celles de stratégie de contrôle en boucle fermée.

La principale limitation de cette stratégie d'asservissement de forme est son coût calculatoire. Même si ce dernier est allégé par l'utilisation d'une fenêtre glissante, l'implémentation de la stra-

tégie demande des réglages : en pratique, la stratégie est calibrée pour être synchronisée à la fréquence de la méthode de perception utilisée. Une seconde limite vient du paramétrage des matrices de poids. Même si le recours à l'apprentissage par démonstration repousse cette limite, il reste à régler des métaparamètres : le nombre de phases de la commande et le comportement des fonctions de mutations.

Certaines questions restent ouvertes. Premièrement, une preuve formelle de la stabilité de la commande. Deuxièmement, les techniques de filtrage les plus idoines pour limiter l'effet des bruits de perception : les pistes envisagées sont les filtres de Kalmann, ainsi que les fenêtres glissantes pondérées. Troisièmement, étudier d'autres formulation des matrices Q et R - transformation spatiale pour décorréler forme et position dans le cas général, ainsi que l'intégration de la rotation finale attendue des préhenseurs dans l'état de l'objet permettrait de poursuivre les améliorations de la stratégie proposée.

La stratégie `OSS` a été intégrée à une l'architecture logicielle `Hades`, puis concrètement testée et comparée à d'autres stratégies dans un contexte industriel. Le cheminement et les résultats liés à cette architecture sont développés dans le chapitre 4.

Chapitre 4

Proposition, intégration et performances d'`Hades`, architecture logicielle dédiée à la manipulation d'objets déformables

Une approche de modèle, méthode de perception ou stratégie d'asservissement de forme, développées indépendamment en laboratoire de recherche, devra être intégrée à un système complet dans le cadre d'un usage industriel. Ces approches, méthodes et stratégies ne reposent pas uniquement sur du matériel : tout le pendant logiciel doit être pris en compte, avec les questions d'intégration, de compatibilité, d'interfaçage et de répétabilité. Il est évident que la solution permettant une fusion des composants logiciels doit permettre une manipulation précise, efficace et adaptable. Il faut donc être capable de s'interfacer avec toute une panoplie de matériels, ainsi qu'avec un simulateur physique, un planificateur de mouvements, éventuellement une interface graphique pour remonter des informations pertinente concernant n'importe lequel de ces systèmes, pour pouvoir intégrer des algorithmes de perception de forme et d'asservissement de forme. Cela pose la question suivante : quels sont les outils logiciels nécessaires pour développer des procédés d'automatisation de manipulation d'objets déformables ?

L'intégration est d'autant plus complexifiée par l'unicité de chaque démonstrateur académique, un logiciel bien trop souvent en écriture-seule suivant l'anti-pattern du tas de boue [FY97]. Cela résulte en une perte de temps, gâchis des moyens humains et techniques mis en œuvre pour les personnes voulant porter un démonstrateur sur leur plateforme, tout cela pour au final recréer des solutions fonctionnant déjà.

Nous nous servirons du projet SoftManBot, mettant en œuvre tous les types d'acteurs concernés par ce problème de manipulation d'objets déformables – chercheurs, équipementiers, intégrateurs, industriels – afin de proposer une solution générique permettant de réutiliser des sources provenant du démonstrateur académique typique en les adaptant à un nouveau cas d'usage, prototyper des modules répondant à de nouveaux besoins métiers, et aller jusqu'à l'industrialisation. La mise en commun de sources et de modules permet de proposer une solution visant généricité et réutilisabilité.

Dans ce chapitre, nous présentons `Hades`, Holistic Architecture for DEformable objects Software, une architecture formée de 4 nœuds ROS. `Hades` est réutilisable du fait de sa composition sous forme de modules, entités auto-contenues de code réutilisables. Chaque nœud est composé d'une partie générique et d'une partie spécifique. La partie générique est un squelette mettant en place la communication internodale et l'appel des fonctions spécifiques. Les fonctions spécifiques sont les drivers et algorithmes implémentés sous forme de modules réutilisables. Ainsi, les utilisateurs peuvent facilement réutiliser leur banc de test pour une expérience similaire – en terme d'actionneurs, de capteurs ou encore d'algorithmes – ou bien réutiliser des modules déjà dévelop-

pés.

Les contributions apportés par ces travaux sont :

1. l'architecture Hades elle-même, conçue pour répondre aux besoins logiciels d'acteurs venant des milieux industriels et académiques.
2. l'adaptation des preuves de concepts développées lors du projet en modules réutilisables et interchangeables.
3. l'analyse comparative et quantifiée des techniques portées dans Hades appliquées à un cas d'usage réel.

Structure du chapitre

Tout d'abord, la section 4.1 explicitera les besoins, détaillera ce que nous entendons par réutilisabilité, et en quoi l'état de l'art ne répond pas à nos problématiques.

Ensuite, nous donnerons une vision globale de haut niveau d'Hades dans la section 4.2.

Nous proposons trois scénarios de validation. Le premier, section 4.3, est un cas de validation simple, utilisant des techniques existantes et maîtrisées, qui nous a permis de tester la structure globale proposée.

Le second, section 4.4, présente les utilisations faites d'Hades dans différents cas d'usage du projet sur le banc de test académique, en intégrant sous forme de module les travaux de recherche effectués dans le cadre du projet.

Le troisième, section 4.5, présente le portage des travaux de recherche sur le banc de test industriel, validant la réutilisabilité des modules développés. Certains modules issus des travaux de recherche étant non satisfaisant pour le cas d'utilisation. Nous avons développé d'autres solutions pour répondre au problème réel, exploitant la modularité d'Hades afin de conserver les développements fonctionnant dans le cas applicatif.

Enfin, la section 4.6 analyse les forces et les faiblesses des différents modules développés. Cette dernière section a un intérêt double. Premièrement, elle nous permet d'avoir une réponse à la question de la meilleure combinaison d'approches de modélisation, de méthodes de perception, et de stratégies de contrôle pour résoudre la tâche. Deuxièmement, elle prouve qu'Hades est un outil permettant de répondre à cette question.

4.1 Formalisation du problème et analyse des besoins

Une tâche robotique industrielle de manipulation d'objets rigides comporte en soi un nombre de challenge importants : comment contrôler les robots, la saisie de l'objet, la localisation de l'objet, l'évitement de collisions, la synchronisation avec le reste de la ligne de production comme par example des tapis roulants. Les objets déformables apportent leur lot de défis spécifiques :

— Il ne faut plus seulement détecter la position de l'objet à manipuler, mais également sa configuration.
— Les points de saisie peuvent dépendre de la dite configuration.
— La saisie elle-même, devant préserver l'intégrité structurelle de l'objet tout en évitant les glissements.
— La gestion de collision est complexifiée.
— Manipuler l'objet déformable implique avoir un modèle et une stratégie d'asservissement de forme.

Toutes ces raisons poussent au développement de nouvelles techniques de perception, de planification, de coopération, de contrôle et d'apprentissage pour la manipulation d'objets déformables.

Un des problèmes identifiés au lancement du projet SoftManBot est l'absence d'outil holistique adapté à l'intégration de telles technologies. La création d'un tel outil devient dès lors un enjeu académique et industriel.

4.1.1 Besoins pour la manipulation robotique d'objets déformables

Une architecture logicielle dédiée à la manipulation d'objets déformables permettrait de tester, itérer, transférer aisément les solutions de perception et d'asservissement de forme entre les personnes intéressées par ces problèmes, et de relever les principaux défis associés à ces derniers, à savoir :

(i) Trouver la combinaison de méthode de perception, d'approche de modèle et de stratégie d'asservissement de forme la plus adaptée pour résoudre une tâche donnée. Toute la complexité vient de la nécessité de rendre compatibles ces trois éléments, fortement dépendants de l'objet déformable manipulé.

(ii) Offrir la possibilité d'itérer sur des solutions existantes, c'est à dire reprendre et modifier des sources pour les adapter à des nouveaux cas d'utilisation.

(iii) Tout en satisfaisant des contraintes molles de temps-réel : les différents algorithmes doivent donner une réponse correcte en un temps court et répétable, pour être programmé sous forme de boucles au pas de temps connu, mais une latence inattendue ne remet pas en question la validité intégrale de la donnée : le maillage perçu une fraction de seconde trop tard, ou une commande légèrement datée seront des résultats supposément meilleurs que ceux du pas de temps précédent.

Nous avons besoin d'un cadre logiciel capable non seulement d'effectuer une tâche de manipulation d'objets déformables, mais également de reprendre une tâche précédemment développée et de l'adapter à un nouveau cas d'usage, en ne modifiant que les parties le nécessitant. L'étude systématique de la littérature [AB16] catégoriserait notre problème comme appartenant à la catégorie "Design et programmation". Le code devra être séparé en modules, unités logiques indépendantes dont l'implémentation exacte doit être abstraite par des interfaces, nos fichier Header en C++. La modularité du code permet sa lisibilité, son encapsulation (éviter d'impacter les fonctionnalités sur lesquelles on ne travaille pas directement), et surtout sa réutilisabilité. Le problème de programmation est défini comme développer et étendre un framework qui permet la réutilisabilité et la modularité de code pour la robotique. Le problème de design est défini comme l'abstraction des modules de code et leurs interconnections aux composants et connecteurs architecturaux. Ceci vient au support des notions de robotique basée composant [Bro+05], concept poussant les chercheurs à créer des composants logiciels, pendant des composants matériels, pour éviter le problème du tas de boue.

Séparation des fonctions logicielles :

La première question à se poser est la définition du système dont l'architecture est le pendant logiciel. Dans le monde physique, se trouve le matériel composant une cellule robotique typique. Elle est composée des robots, des préhenseurs, et éventuellement de toute autre machinerie nécessaire à la tâche. Il y a également les divers capteurs, d'une part les capteurs proprioceptifs, mesurant des métriques relatives aux robots et préhenseurs sus-mentionnés (encodeurs, capteurs de force, mesures de retour d'efforts), d'autre part les capteurs exteroceptifs, mesurant des caractéristiques d'objets autres que les robots : caméras, laser, capteurs tactiles au contact de l'objet déformable. L'objet déformable en question fait bien sûr parti de l'environnement, tout comme un éventuel opérateur humain.

Dans le monde logiciel, nous devons tout d'abord nous interfacer avec le monde physique. Tous les capteurs et actionneurs sus-mentionnés doivent être intégrable dans l'architecture. Il faut également que l'opérateur humain puisse agir sur le système par le biais d'une Interface Homme

Machine (IHM). Les données des capteurs seront filtrées par des algorithmes de fusion de donnée, pour reconstruire la forme actuelle de l'objet. Il faut également incorporer des stratégies d'asservissement de forme, permettant de générer les ordres des robots. Enfin, il est nécessaire d'avoir une maîtrise haut-niveau du système pour coordoner les diverses opérations, transmettre les ordres reçus de l'IHM, et intégrer des stratégies de haut niveau, comme de la reprise sur erreur, ou la gestion du système global grâce à une IA.

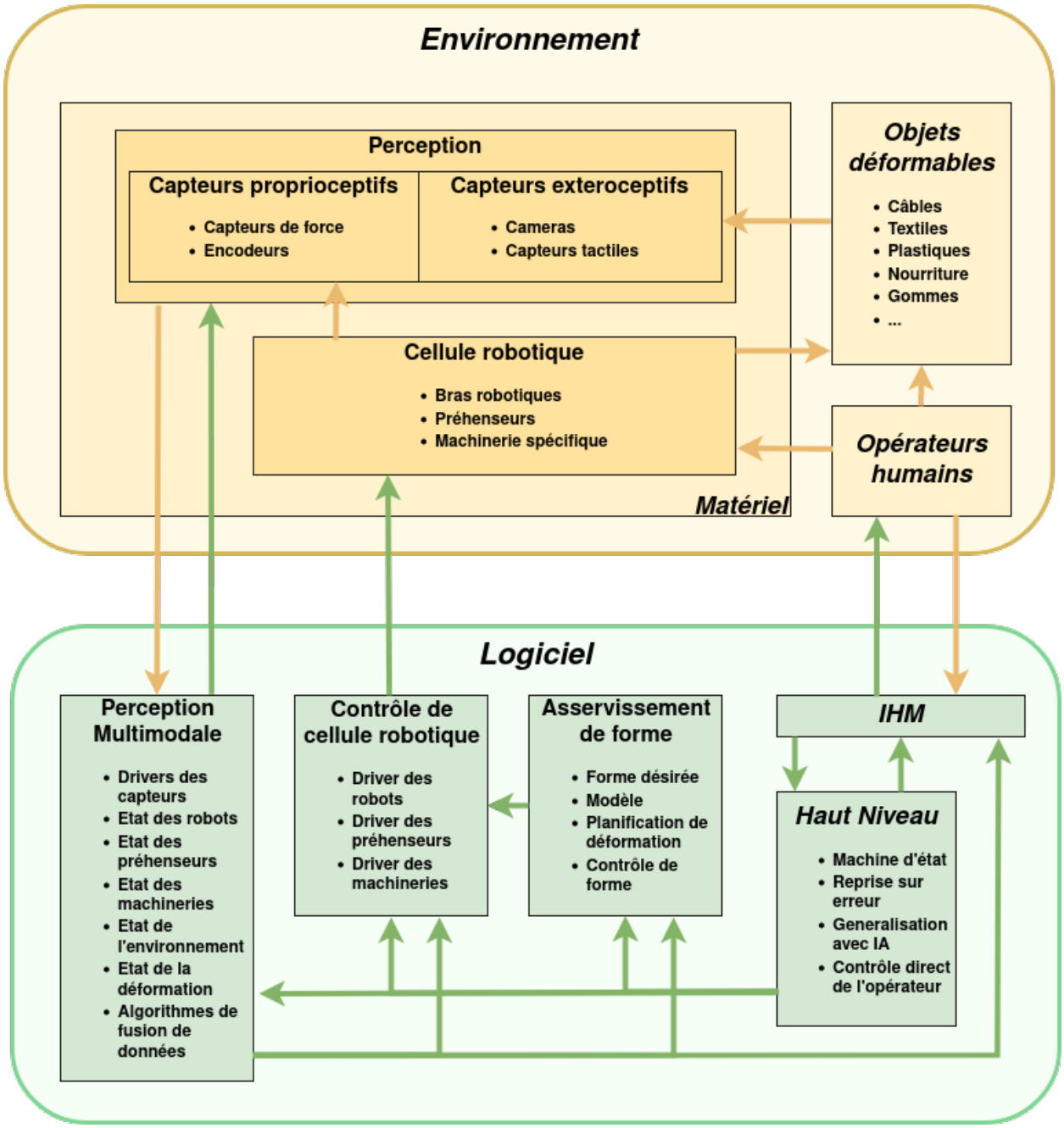

FIGURE 4.1: Représentation modulaire de l'architecture matérielle et logicielle générique

Il reste à convenir comment séparer le code pour permettre la modularité souhaitée. Logiquement, une stratégie d'asservissement de forme doit être utilisable peu importe quelle méthode de perception est utilisée, pourvue que les différentes méthodes retournent des objets de même type. De même, les stratégies de contrôle de forme doivent fonctionner sans être influencées par la configuration matérielle exacte du système, par exemple la marque des robots et préhenseurs utilisés.

Enfin, la solution d'implémentation de la machine d'état – piloté par PLC, par IA, par heuristique, ou par IHM, est indépendante de tous ces autres points. Un avantage d'un tel découpage en module est la possibilité de parallèliser les traitements : une méthode de perception pourra évoluer dans un programme ou thread (fil d'exécution) indépendant de la stratégie d'asservissement de forme, permettant de bénéficier du nouveau standard multi-cœurs des processeurs modernes.

Pour résumer,

— L'environnement comporte les objets déformables, les opérateurs humains, et le matériel, lui-même composé des actionneurs formant la cellule robotique et des capteurs.

— D'un point de vue logiciel, les fonctions sont séparable en 4 blocs : la perception multi-modale, le contrôle de la cellule robotique, l'asservissement de forme et la gestion haut niveau, interfacé avec un opérateur humain par une IHM. Il est souhaitable que ces blocs soit exécutés de manière parallèle.

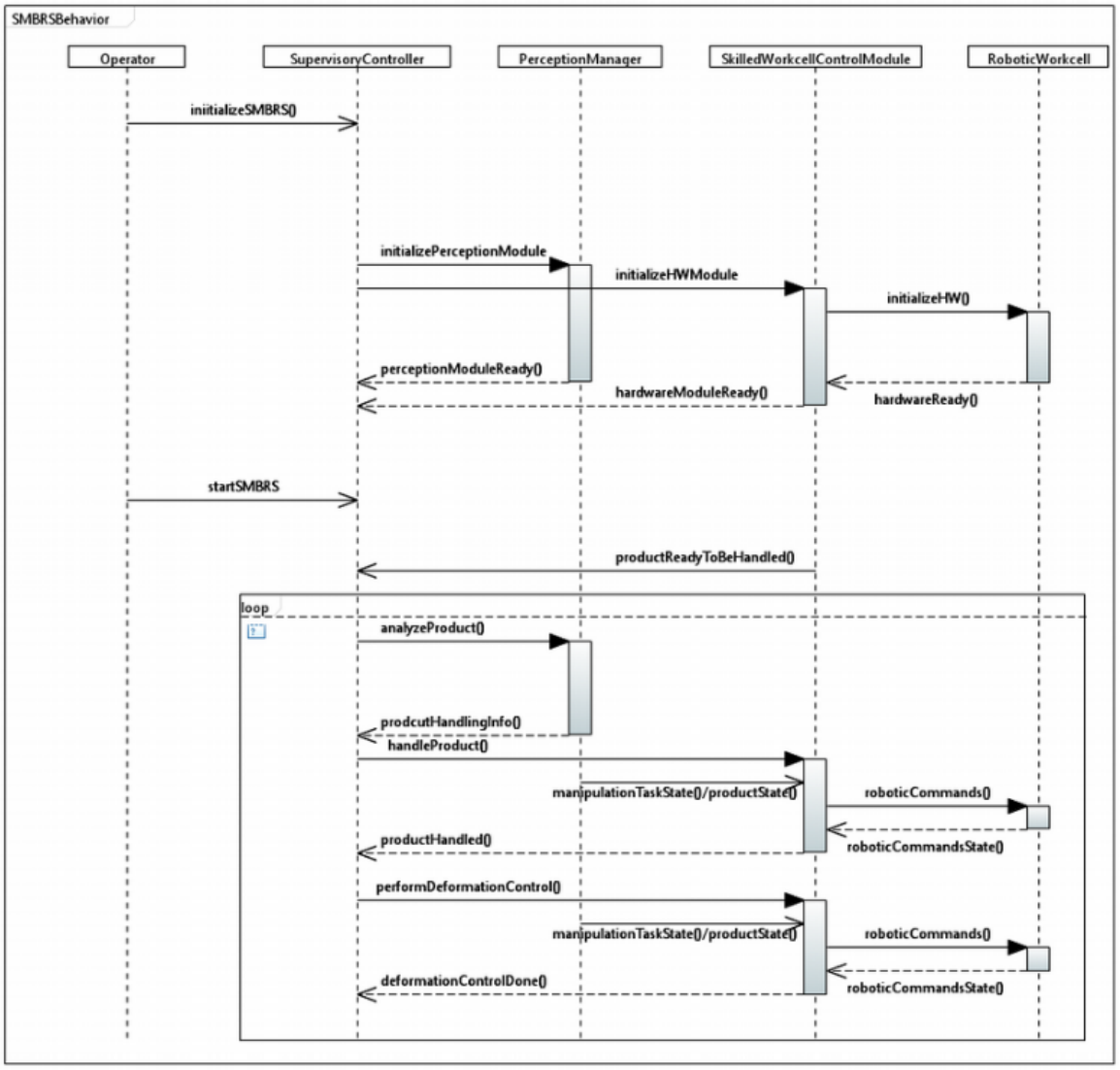

FIGURE 4.2: Description de la séquence d'exécution souhaitée par un système logiciel de manipulation d'objet déformable

extrait du déliverable SoftManBot "Global Definition of the Software and Hardware Architectures"

Cette représentation est montrée dans la Fig. 4.1, donnant une première vision de la struc-

ture logicielle nécessaire. De même, les utilisateurs finaux interrogés ont convenu d'une séquence d'exécution typique présentée dans la Fig. 4.2.

Nous avons donc besoin d'une base de code permettant de mettre en place des interfaces et de brancher/débrancher aisément nos modules.

En plus de la modularité souhaitée et de la réutilisabilité des exécutables, la solution doit suivre plusieurs critères liés au projet. Ces critères sont :

A - Être suffisamment flexible pour le développement Agile, avec des itérations rapides et des retours d'expériences entre expérimentations et spécifications de modules.

B - Ne pas dépendre d'une licence commerciale

C - Utiliser ROS, le standard de-facto pour l'intégration de systèmes robotiques. En particulier, le choix a été arrêté sur ROS-1 version Noetic. Fixer une version permet une visibilité à long-terme pour les partenaires industriels.

4.1.2 Réutilisabilité

La base de code se doit d'être réutilisable, dans tous les sens du terme. Il y a 2 types de réutilisabilité dans le développement logiciel. Le premier est de faire tourner un logiciel exécutable existant malgré des changements environnementaux, comme les librairies annexes installées ou la version de l'OS utilisée. Le second type de réutilisabilité est la modification d'une base existante de sources de code pour résoudre un nouveau problème, connexe à un ancien déjà résolu.

Réutilisabilité des exécutables

Trois choix s'offrent au développeur.

— Linker statiquement un exécutable avec ses bibliothèques. C'est la version la plus simple pour l'utilisateur, qui n'a qu'à lancer le programme. Cependant, c'est aussi la version la plus gourmande en mémoire, car les applications ne partagent plus leurs dépendances : cela pose des problèmes de maintenance à long terme car une mise à jour d'une dépendance nécessite de re-linker l'application du côté du programmeur, et de la re-télécharger du côté de l'utilisateur.

— Linker dynamiquement l'exécutable avec ses bibliothèques et les gérer avec un gestionnaire de paquet. Cela permet une bien meilleure gestion et maintenance, tout en évitant la prolifération des duplications de bibliothèques sur une machine. Cependant, cela expose l'utilisateur à de chronophages conflits de dépendances. La gestion des bibliothèques peut se faire grâce à des gestionnaires, comme pip pour python ou Conan pour le C++.

— Conteneuriser l'exécutable. En paramétrant une machine virtuelle individuelle contenant uniquement le minimum pour qu'une application fonctionne, le code pourra continuer de fonctionner pendant des années malgré les évolutions, et absences d'évolutions, de bibliothèques en restant auto-compatible. Cela nécessite cependant plus de mémoire et de puissance que les solutions précédentes. Dans le cas de la robotique, la solution consensuelle consiste à utiliser Docker [WC17], dont le fonctionnement est bien aligné avec la base de code ROS.

Ces choix sont résumés dans le tableau 4.1.

TABLE 4.1: Solutions de réutilisabilité pour exécutables logiciels

Solutions	**Avantages**	**Inconvénients**
Linkage statique	Simple pour l'utilisateur	Maintenance compliqué
Linkage dynamique	Maintenance simple Plus efficace en mémoire utilisée	Le plus complexe pour l'utilisateur
Conteneur	Le plus pérenne	Le plus gourmand en mémoire

Nous avons choisi dans le cadre de la thèse la solution du linkage dynamique, permettant facilement de reprendre un code d'un cas d'usage et de le transférer entre membres du projet en récupérant uniquement les fonctions d'intérêt.

Vu le faible nombre d'utilisateurs et le fort besoin d'adaptabilité à chaque nouveau cas d'usage, il n'est pas pertinent de regrouper l'application développée sous forme d'un exécutable UNIX disponible via commande apt.

Réutilisabilité des sources Le second type de réutilisabilité est la réutilisabilité des sources. Cet enjeu est capital dans le projet, pour permettre la transmission et la récupération de solutions de contrôle de forme, de perception, de drivers bas niveau ou encore d'IHM parmi les partenaires, tout en permettant la flexibilité et l'adaptation à une nouvelle cellule robotique ou un nouveau type d'objet déformable. Cet objectif est généralement atteint dans le paradigme Orienté Objet grâce à l'utilisation de modules, des unités de codes bien définies effectuant une fonction spécifique avec des interfaces fixées.

Le partage des sources permet de recompiler un scénario fonctionnel, avant d'échanger les modules pertinents ou la configuration afin de l'adapter au cas d'usage sur lequel l'architecture est appliquée.

4.1.3 Positionnement relatif à l'état de l'art

Dans la littérature, plusieurs architectures logicielles ont été proposées pour la robotisation des processus industriels, comme détaillé dans la section 2.4. En dehors des solutions très spécifiques à un cas d'usage, les propositions suivantes ont retenu notre attention :

•*Robmosys* [1] est un framework imposant un travail en cycle en V – rentrant en contradiction avec le critère **A**.

•*skiROS* [Rov+17] est un framework similaire à Robmosys – rentrant en contradiction avec le point **A**. skiROS est soumis à des licences limitant l'usage commerciale – rentrant en contradiction avec le point **B**.

•*PlanSys2* [Mar+21] est le successeur libre de skiROS – rentrant en contradiction avec le point **A** – pour ROS2 – rentrant en contradiction avec le point **C**.

Cette analyse est résumée dans le tableau 4.2.

1. https ://robmosys.eu/

TABLE 4.2: Table de comparaison de l'état de l'art des architectures

Architecture	A : Agilité	B : Liberté	C : ROS-1
Robmosys	X	✓	✓
skiROS	X	X	✓
PlanSys2	X	✓	X

Ainsi, aucune solution disponible présentée précédemment ne correspond à nos besoins : de plus, aucune architecture générique disponibles n'est dédiée à la problématique de l'objet déformable. Par conséquent, nous avons décidé de développer notre propre solution tout en la rendant adaptable aux applications générales de manipulation robotique d'objets déformables.

Enfin, le problème est proche des problématiques temps réel. Le temps réel est défini comme l'assurance de délivrer une réponse en un temps fixé à l'avance. Cela nécessite que l'OS sur lequel le code fonctionne offre lui-même les garanties d'atteindre des objectifs en des temps bornés. Le vrai temps réel n'est pas possible étant donné les exigences du projet. Cette limitation est due au choix de ROS-1. En effet, le vrai temps réel est supporté depuis ROS-2 uniquement [Fis+19].

D'un point de vue fonctionnel, les contraintes dures de temps réel ne sont pas obligatoires. En particulier, un OS non certifié temps réel n'offre aucune garantie vis à vis de la préemption, c'est à dire l'interruption d'une tâche par l'ordonnanceur par une autre tâche jugée plus prioritaire. Malgré cela, un programme suffisamment rapide délivrera des ordres aux robots à une fréquence suffisante à l'accomplissement de la tâche, même si l'un de ces ordres se trouve retardé du fait de la gigue du système - c'est à dire la variation du délai alloué à une même tâche par l'OS entre deux appels.

4.2 Proposition d'architecture logicielle : `Hades`

En réponse à l'absence de solution convenant à notre problème, nous avons décidé de développer notre propre architecture, `Hades`. Cela nous permet d'avoir la main sur l'intégralité du procédé de développement, et facilite donc l'implémentation, le transfert et le suivi de bug. `Hades` est utilisée pour des cas d'usages de SoftManBot et est, par extension, adaptable pour d'autres cas impliquant de la manipulation d'objets déformables.

4.2.1 Structure haut niveau : tâches et skills

Par définition dans le projet, les cas d'usages ont été divisés en tâches, elles-mêmes sous-divisées en une ou plusieurs skills. Une tâche représente l'action qu'on veut effectuer, une skill est une opération élémentaire à associer pour accomplir une tâche. En particulier, une skill peut être employée plusieurs fois pour des tâches différentes. Ces skills sont implémentées sous forme de modules interchangeables.

Le code développé fournit des interfaces pour implémenter 3 tâches différentes, faites de 3 skills. En cas de besoins supplémentaires, il est très facile d'extrapoler les interfaces à définir pour étendre les capacités des interfaces. Avec le recul obtenu lors de nos premiers tests, les tâches sont subdivisées en des fonctions "Grasping" et "Perform". En effet, la saisie de l'objet déformable est un problème tout aussi complexe que la manipulation du dit objet déformable. De plus, nous avons ajouté des tâches génériques.

— La tâche `Idle` permet de mettre le système en stand-by, ou éventuellement d'afficher des informations de debug. Cette mise en pause est utile par exemple pour manipuler l'objet

sans créer d'erreur dans un algorithme de perception peu robuste en cours de développement.
- La tâche `Stop` permet d'arrêter proprement les différents algorithmes en leur indiquant que la séquence en cours est terminée, ainsi que de dégager les robots avec des trajectoires adaptées à la fin de tâche.
- La tâche `Init` permet de réinitialiser l'état du logiciel : par exemple, les positions initiales des algorithmes de suivi de forme sont recalculées, les erreurs intégrales sont purgées, et les robots sont envoyés dans une configuration articulaire maîtrisée.

Ces tâches génériques sont indispensables au fonctionnement et présentent plus qu'un intérêt pratique lors des tests : par exemple, l'enchaînement Stop/Init permet une meilleure répétabilité lors de l'enchaînement multiple de la tâche en permettant de forcer les configurations aussi bien logicielles que mécaniques.

Les utilisateurs souhaitant mettre en place leur cas d'usage ont à disposition un squelette contenant une structure prête à accueillir des nouveaux développements. Les sources peuvent être divisées en 3 catégories :
- Les sources génériques, qui contiennent les définitions des nœuds et de l'intercommunication nécessaire au fonctionnement de la base de code.
- Les sources spécifiques, dans lesquelles l'utilisateur va mettre en place la colle logicielle, c'est à dire le code appelant les modules via leurs interfaces.
- Les modules, récupérés dans une base de modules existants ou développés par l'utilisateur.

4.2.2 Description d'`Hades`

Suite à la définition des besoins, nous souhaitons donc utiliser une structure logicielle composée de 4 entités interconnectées comme dans le schéma montré Fig. 4.3.

`Hades` a été implémentée sous forme de 4 nœuds ROS, – nommés respectivement `supervisor`, `skilledWorkcell`, `perception` et `deformationControl`. Chaque nœud est donc un programme indépendant, évoluant dans un thread séparé avec son propre espace mémoire. Ceci permet une bonne parallélisation et un interfaçage facile avec les composants basés sur ROS, ainsi que l'accès aux services, actions, topics et messages. Cependant, le debug de programmes multithreadés est notoirement difficile, et ROS-1 ne propose pas d'outil natif avancé et ergonomique de débug.

sources communes Une partie des sources est dans un répertoire commun. Ce répertoire comporte les sources suivantes :
- Les parties génériques des nœuds. Elles ont été placées dans ce répertoire pour que les répertoires correspondant aux nœuds ne soient pas encombrés de fichiers que l'utilisateur ne doit pas modifier.
- `genericLogic.hpp`, contenant deux énumérations. La première est l'énumeration des états possibles de la machine d'état, la seconde celle des ID des nœuds. En utilisant les mots-clef `typedef enum`, nous pouvons avoir un raisonnement fortement typé, ce qui impose des contraintes sur la manipulation de données, améliorant la robustesse du code en permettant une détection précoce des erreurs à la compilation plutôt qu'à l'exécution. Les états et les ID des nœuds ne sont plus des nombres, mais des types de variable, ce qui clarifie le code et le raisonnement. Ce nœud contient également une définition de la classe "pose", exprimée en tuple (x,y,z,roll,pitch,yaw). Nous avons créé 3 constructeurs – par défaut, position et orientation – qui couvrent nos cas d'usage. Nous avons également introduit des surcharges des opérateurs d'addition, de soustraction de pose entre elles ainsi que de multiplication et division par des flottants ce qui simplifie la lecture des algorithmes de plus

haut niveau, en particulier ceux de contrôle de déformation. Cette "pose" peut exprimer, selon le contexte, une position absolue ou une vitesse. Enfin, cette classe est sérialisable pour être encapsulée dans la communication internodale plus facilement.

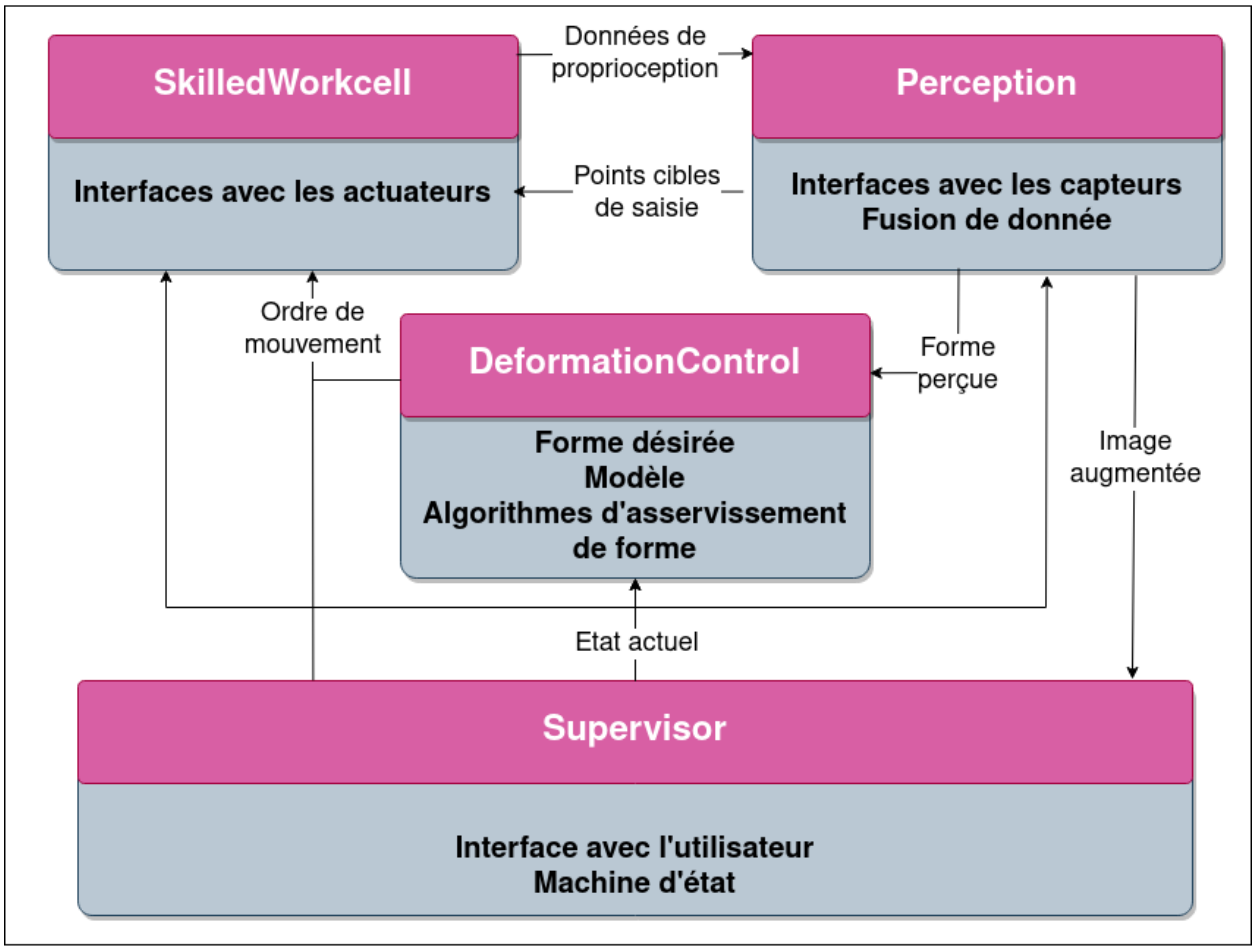

FIGURE 4.3: Schéma bloc d'Hades

— `mesh.cpp`, définissant des classes nodes, triangle et mesh. Un mesh est formé d'un vecteur de nodes et d'un vecteur de triangles. Cet objet permet de représenter l'état de nos objets déformables en se basant uniquement sur la connectivité des éléments. Cet objet est sérialisable – convertible en un format séquentiel pour être transféré ou stocké, au format json car la lisibilité par l'humain a sa pertinence pour les positions courantes et positions cibles : par exemple, cela permet la détection d'une rotation de 180° d'un objet texturé entre position détectée et position cible sans devoir risquer une expérience ou écrire un code de détection. La classe est définie dans le namespace – sous-domaine de noms de variables – softmanbot pour éviter les collisions de noms avec l'une des nombreuses bibliothèque redéfinissant les concepts de node, triangle ou mesh avec des implémentations différentes. Ces derniers seront donc des variables de type softmanbot : :node, softmanbot : :triangle ou softmanbot : :mesh.
— `positionConverter.cpp`, contenant des fonctions utilitaires permettant de transformer des orientations exprimées en quaternion ou Roll-Pitch-Yaw, transformer des positions exprimées en geometric_message de ROS ou dans les types définis par les utilisateurs, encapsuler les fonctions de transformation de la bibliothèque ROS tf, effectuant des transformées dans l'espace.

— Les bibliothèques logicielles header-only linkées à plusieurs nœuds, comme nlohmann : :json, permettant de sérialiser et désérialiser des formats json, et ZMQ, une bibliothèque de communication réseau légère implémentant des sockets dans toute une variété de langages de programmation, rendant cette communication agnostique au langage utilisé.

.

supervisor Le nœud `supervisor`, comme son nom l'indique, supervise les opérations. Ce nœud a la responsabilité de gérer la machine d'état, émettant l'état actuel aux autres nœuds, comme présenté dans l'algorithme 6. Il prend en compte pour cela l'état d'avancement des autres nœuds, sachant ceux qui ont ou non fini la tâche actuelle, état sauvegardé dans la variable booléenne *isStepDone*.

L'implémentation de la fonction `getState()` est spécifique à l'application : on peut la piloter avec une logique interne, un PLC pour permettre une meilleure sécurité des utilisateurs et du matériel, ou une interface utilisateur comme un terminal ou une Interface Homme Machine.

Algorithme 6 Machine d'état

```
1:  if isStepDone then
2:      previousState = currentState
3:      currentState = getState()
4:      switch currentState do
5:          case 0
6:              if previousState != IDLE then
7:                  previousState= IDLE
8:              end if
9:          case 1
10:             if previousState != INIT then
11:                 previousState = INIT
12:                 isStepDone = false
13:             end if
14:         case 2
15:             if previousState != GRASPINGT1S3 then
16:                 previousState = GRASPINGT1S3
17:                 isStepDone = false
18:             end if
19:         case 3
20:             if previousState != PERFORMT1S3 then
21:                 previousState = PERFORMT1S3
22:                 isStepDone = false
23:             end if
24:         case 4
25:             if previousState != STOP then
26:                 previousState = STOP
27:                 isStepDone = false
28:             end if
29: end if
```

C'est également le nœud qui a pour rôle d'interagir avec l'opérateur, que ce soit pour prendre des ordres ou remonter des informations. Ce rôle est rempli en utilisant une Interface Homme Machine, montrée dans la Fig. 4.6.

perception Le nœud `perception` gère les capteurs, de l'intégration du hardware à la reconstruction de l'état de l'objet, en passant par les algorithmes de fusion de données. Tous les travaux de perception ont été faits dans le repère de la caméra, sans transformation ou autres changements de repère, et transmis dans ce même repère caméra. En effet, il est à la fois moins coûteux en calcul et plus intuitif de transformer deux positions de robots que l'intégralité d'un nuage de points, ou encore un réseau d'une centaine de nœuds. La pertinence de ce choix peut bien sûr être remise en question, par exemple dans le cas d'usage de Plastinher où il y a 2 caméras et un seul robot – un seul robot est suffisant pour extraire une semelle de chaussure, mais la vision frontale et latérale du système est désirable.

A part le cas de l'objet linéique, aisément représentable par un vecteur sérialisable de positions, la perception est sensée détecter un objet au format softmanbot : :mesh. Le nœud qui exploitera ces résultats, `deformationControl`, a accès aux positions des nœuds ainsi que leur interconnexion via les triangles. Il est également possible d'extraire des structures de structures de plus haute dimensions, typiquement extraire un objet surfacique d'un objet volumique, ou bien un objet linéique d'un objet surfacique.

skilledWorkcell Le nœud `skilledWorkcell` gère toute la machinerie, en particulier les contrôleurs bas-niveau des robots, les préhenseurs et tout autre type d'actionneurs dédiés à la tâche. Tout nœud nécessitant des informations ou des interactions avec du matériel le fait par le biais de services, que ce soit pour transmettre des ordres aux bras robotiques, actionner les préhenseurs ou communiquer avec le PLC. Les ordres envoyés aux robots sont exprimés dans le repère de la caméra, c'est donc à ce nœud de gérer les changements de repère.

deformationControl Le nœud `deformationControl` gère le contrôle haut-niveau, c'est à dire la gestion de la forme de l'objet. C'est ce nœud qui calculera les ordres à donner au robot. Selon la stratégie d'asservissement de forme utilisée, les données provenant de la perception permettront de reconstruire un modèle et ainsi générer une trajectoire de déformation, avec les ordres à donner aux effecteurs des robots.

Procédé de compilation Le code est écrit en C++, version 17. Le choix de la version est celle la plus avancée compatible avec ROS Noetic, car le C++-20 est connu pour créer des problèmes avec les versions Melodic et Noetic. Ce dernier point est regrettable, car l'utilisation de coroutines aurait été appréciable, en particulier pour la communication avec les grippers via le PLC. Le code est linké dynamiquement. Cette décision est basée sur les dépendances communes à certaines bibliothèques externes comme Eigen.

Nous avons séparé les fichiers CmakeList. Le CmakeList global s'occupe d'appeler les fichiers spécifiques de recherche de bibliothèques, de chercher les bibliothèques ROS communes, de générer les messages et services, et enfin d'appeler les fichiers spécifiques de linkage. Ceci permet de cibler précisément les dépendances de chaque nœud, afin d'améliorer la maîtrise et la clarté de la gestion de bibliothèques.

4.2.3 Interfaces

Interfaces intra-nodales :

Nous avons utilisé le design pattern "Interface" pour les interfaces intra-nodales. Les fichiers génériques définissent dans les header des interfaces, implémentant constructeurs vides mais surtout un éventail de fonctions reflétant les différentes tâches définies en amont du projet. Par défaut, ces fonctions virtuelles renvoient une exception "non implémentée", comme montré dans la Fig.

4.4. L'utilisateur définit dans ses sources spécifiques une classe héritant de l'interface, redéclarant (et redéfinissant dans les fichiers spécifiques) les fonctions qui seront appelées après traitement de la machine d'état. Dans le cas où la fonction n'est pas implémentée, le design de l'interface intra-nodale fera remonter une exception, nous assurant de l'adéquation entre implémentation de la machine d'état et fonctions spécifiques au cas d'usage.

```
class supervisoryInterface
{
public:
  supervisoryInterface(void) {};
  ~supervisoryInterface(void) {};

  virtual void supervisoryInit(void) {throw std::string("exception : Not implemented");};
  virtual void supervisoryIdle(void) {throw std::string("exception : Not implemented");};
  virtual void supervisoryStop(void) {throw std::string("exception : Not implemented");};

  virtual void supervisoryGraspingT1S1(void) {throw std::string("exception : Not implemented");};
  virtual void supervisoryPerformT1S1(void) {throw std::string("exception : Not implemented");};
  virtual void supervisoryGraspingT1S2(void) {throw std::string("exception : Not implemented");};
  virtual void supervisoryPerformT1S2(void) {throw std::string("exception : Not implemented");};
  virtual void supervisoryGraspingT1S3(void) {throw std::string("exception : Not implemented");};
  virtual void supervisoryPerformT1S3(void) {throw std::string("exception : Not implemented");};
};
//Custom declaration of inherited interface
class michelinSupervisoryInterface: public supervisoryInterface
{
public:
  void supervisoryInit(void) override;
  void supervisoryIdle(void) override;
  void supervisoryStop(void) override;

  void supervisoryPerformT1S2(void) override; //Control from IHM
  void supervisoryGraspingT1S3(void) override; //Precise joining of 2 layers
  void supervisoryPerformT1S3(void) override;
};
#endif //GENERIC_SUPERVISOR_HPP
```

FIGURE 4.4: Interface entre parties génériques et spécifiques

Les interfaces sont définies dans des fichiers différents. La classe virtuelle pure a été tronquée des tâches déclarées mais non utilisées pour augmenter la lisibilité.

Interfaces internodales :

La communication entre les nœuds est effectuée avec les topics et services ROS. Pour les objets complexes, nous utilisons le type string contenant l'objet sérialisé avec la bibliothèque boost : :serialization [2]. Fig. 4.5 présente le graph ROS typique d'une application basée sur `Hades`.

— Topic **/supervisor_task** : Ce topic diffuse la tâche actuelle de la machine d'état du nœud `supervisor`. Le type du topic est de type Int8, la correspondance entre nombre et logique est définie dans les fichiers communs. Les fichiers génériques des nœuds `perception`, `deformationControl` et `skilledWorkcell` sont abonnés au topic ce qui permet aux nœuds d'exécuter la tâche associée à l'état.

— Topics **/per_taskDone**, **/sw_taskDone** et **/dc_taskDone** : Ces trois topics, dont le message est un booléen, sont émis par les parties spécifiques des nœuds lorsque la skill en cours est achevée, permettant à la machine d'état de fonctionner en boucle fermée.

2. `https://www.boost.org/`

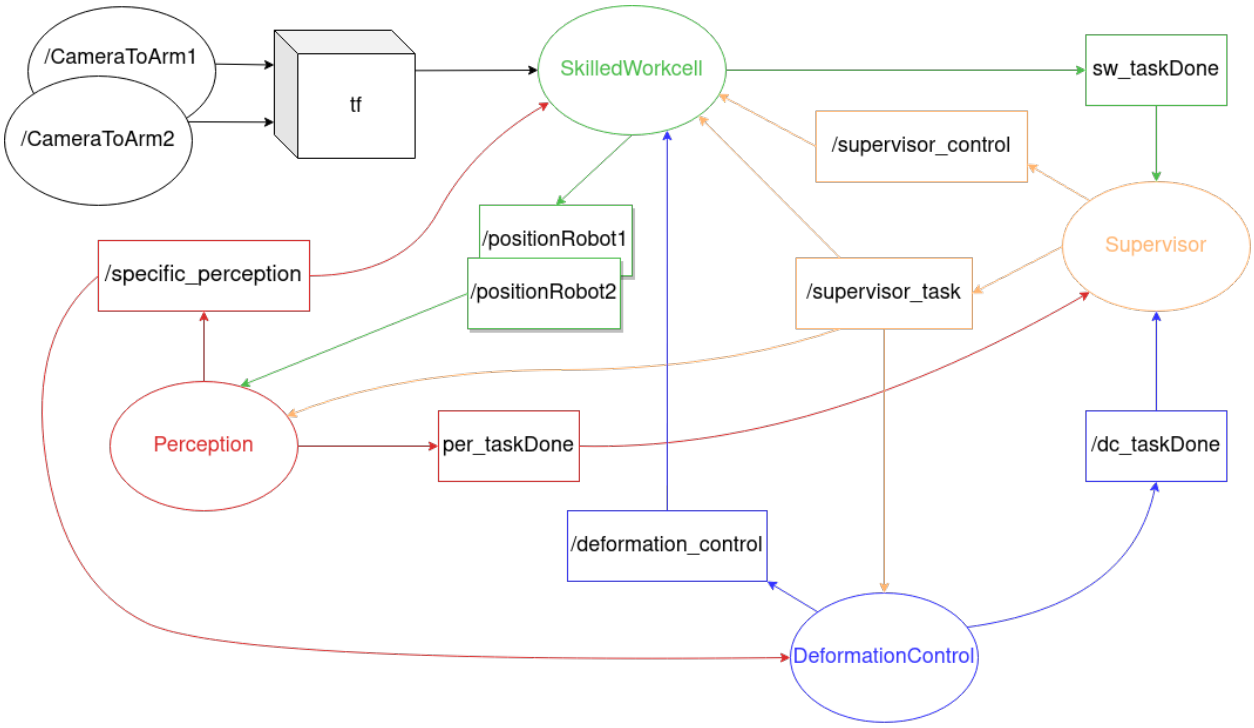

FIGURE 4.5: Graph ROS d'Hades en fonctionnement

Les couleurs correspondent aux nœuds émettant les topics.
Jaune : Superviseur. Rouge : Perception. Vert : SkilledWorkcell. Bleu : DéformationControl

- Topics **/CameraToArm1** et **/CameraToArm2** : Ces deux topics du paquet tf contiennent les transformations statiques entre bras et caméra, paramétrés avec le paquet Moveit Calibration [3]. Tous nos cas d'usages utilisent une seule caméra et un ou deux bras, nous avons donc choisi de raisonner systématiquement en point de vue caméra. Les conversions de positions se font donc uniquement dans le nœud `skilledWorkcell`. Ceci relève du choix et du cas d'usage, non pas du dogme. Implémenté dans les parties spécifiques, il est possible à un utilisateur de choisir un paradigme différent.
- Topics **/positionRobot1** et **/positionRobot2** : Ces topics contiennent les données de position des robots, exprimées dans le repère caméra.
- Topic **/specific_perception** : Ce topic de type string contient les objets sérialisés correspondant à toutes les informations dont ont besoin les autres nœuds : forme détectée de l'objet, informations de proprioception des robots, points cible de saisie... Il est diffusé au nœud `deformationControl` pour fermer la boucle d'asservissement et au nœud `skilledWorkcell` pour lui communiquer les points cibles de saisie.
- Topics **/supervisory_control** et **/deformation_control** : Ces topics permettent respectivement aux nœuds `supervisor` et `deformationControl` d'envoyer des ordres de déplacement, exprimé dans le repère caméra, effectués par le nœud `skilledWorkcell`. Ils sont représenté séparément car le `supervisor` a également accès aux préhenseur, en temps que service.

Interface utilisateur :

L'interface utilisateur prend la forme d'une Interface Homme Machine, montrée dans la Fig.

3. https://ros-planning.github.io/moveit_tutorials/doc/hand_eye_calibration/hand_eye_calibration_tutorial.html

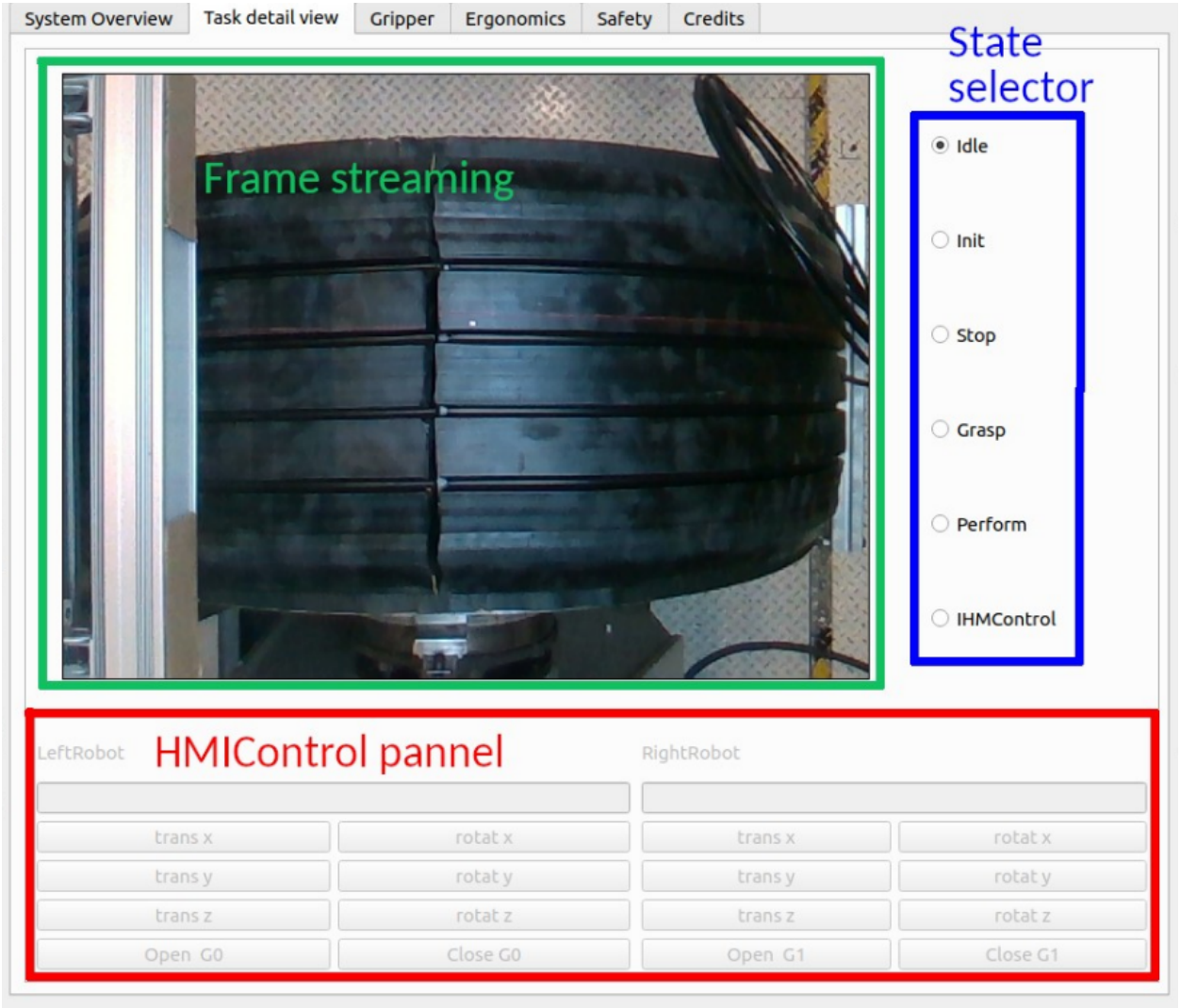

FIGURE 4.6: Interface Homme Machine utilisée pour les cas d'usages

4.6, ainsi que du terminal qui nous permet de remonter des informations au format texte. Comme détaillé plus haut, les applications de manipulation d'objets déformables impliquent souvent l'usage de caméra pour percevoir la forme de l'objet. L'IHM doit donc permettre de visualiser ce que voit le système, en terme de données sensorielles et d'interprétation de reconstruction. L'opérateur peut souhaiter prendre la main sur le fonctionnement du système, pour l'arrêter ou le réinitialiser de manière logicielle : il faut donc que l'IHM puisse contrôler la machine d'état. Enfin, l'opérateur peut souhaiter manipuler les robots ou les pinces directement depuis le poste de contrôle.

La communication avec l'IHM passe par des sockets ZMQ, pour ne pas imposer l'utilisation de ROS à l'équipement utilisé pour le monitoring, seulement une connexion réseau à la cellule SoftManBot. L'IHM montre les frames augmentées venant de la caméra, donne la main sur les changements d'état avec des Radio Button, et permet d'envoyer des ordres de déplacements, rotation, fermeture et ouverture des effecteurs à des fin de prototype. L'intérêt est de garder une interface utilisateur unique malgré des changements potentiels de robots et de préhenseurs - qui ne sont pas eux mêmes obligatoirement regroupés sur une même interface, surtout s'ils proviennent de fabriquant différents.

Interface matérielle :

Le matériel, qu'il s'agisse des bras robots, des pinces, des capteurs, peut être intégré dans

Hades de la manière la plus libre possible grâce à l'abstraction permise par les nodes. En particulier, nous avons implémenté des contrôleurs ROS et des drivers direct pour tous les éléments mentionné ci-dessus, permettant d'obtenir le meilleur des deux mondes et de choisir les compromis qui nous intéressaient le plus au cas par cas.

Interface avec d'autres logiciels : Dans certains cas, nous voulons nous interfacer avec des bibliothèques d'autres langages ou d'autres exécutables. Nous avons utilisé pour ce premier problème de l'embedding - appel de fonctions Python depuis le C++, encapsulé sous forme de fonctions C++ accessibles depuis un header, rendant le changement de technologie transparent à l'utilisateur. Dans le second cas, nous utilisons la bibliothèque réseau fortement agnostique ZMQ, permettant de créer des canaux de communication entre n'importe quels langages de programmation .

4.3 Cas simplifié de validation

Nous avons tout d'abord testé les sources communes et la structure d'Hades sur un cas de validation aussi simple que possible : la régulation de la tension dans une bande de gomme. La bande de gomme peut être assimilée à un objet déformable linéique. Une partie de la bande est tenue par un robot UR10, l'autre est enroulée autour d'une bobine. Cette dernière permet d'induire des perturbations dans la tension de la gomme, en donnant du mou ou tendant le produit, comme montré Fig. 4.7. Le seul capteur utilisé pour cette expérience est une caméra Intel Realsense.

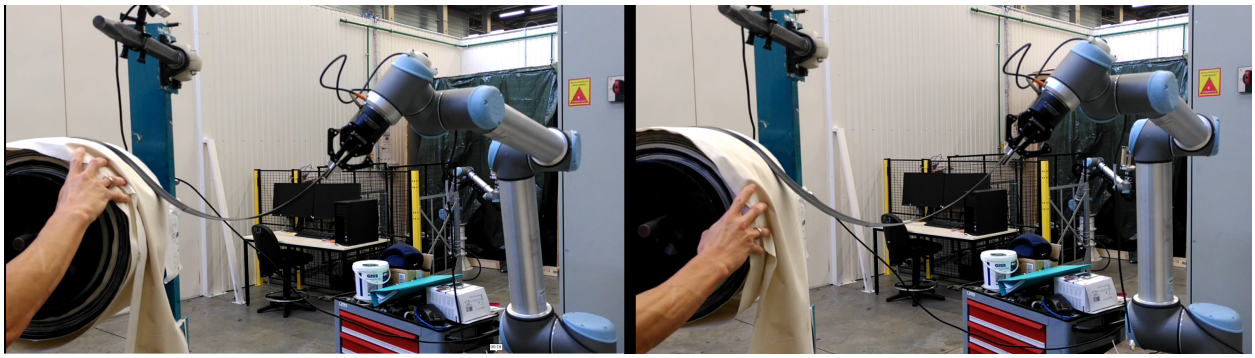

FIGURE 4.7: Vue de la tâche robotique lors du cas de validation

Une vidéo de la tâche est disponible sur https://youtu.be/ZAIfEuWaxGQ

Le cas d'usage a été intégré dans Hades comme schématisé sur la Fig. 4.8. Cette section présente les détails d'implémentation et les choix effectués pour décomposer cette tâche simple, afin d'illustrer l'architecture proposée dans la section précédente.

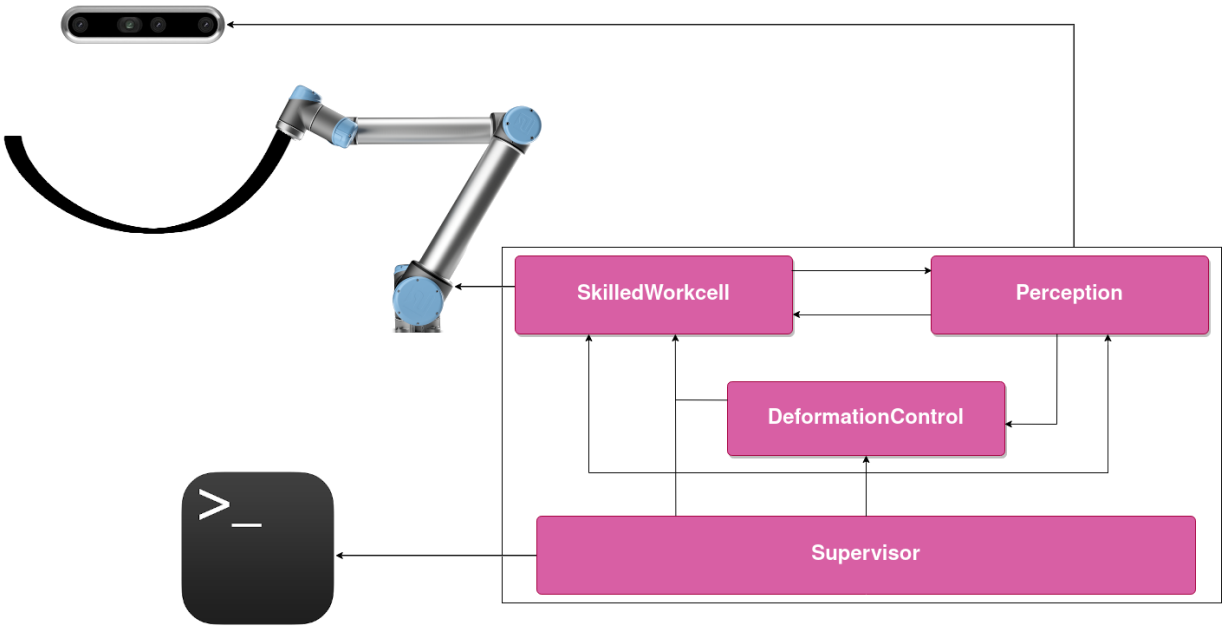

FIGURE 4.8: Configuration du cas de validation

4.3.1 Structure haut niveau

Deux tâches sont utilisées, chacune possédant uniquement une seule skill. La tâche générique `Init` permet de déplacer le robot dans l'espace articulaire afin d'obtenir une position de départ répétable, sans risquer de singularités. La tâche `Control` effectue la régulation.

4.3.2 Nœud supervisor

Ce nœud n'a presque pas de fonctions spécifiques liées au cas d'usage, tout juste de quoi basculer entre les deux tâches. Le `supervisor` déclenche la tâche `Init`. Ensuite, il attend la confirmation de fin d'initialisation de la part des autres nœuds, avant de boucler sans fin sur la tâche `Control`.

4.3.3 Nœud perception

Dans ce cas, le nœud `perception` est celui qui a le fonctionnement le plus complexe. En réutilisant les travaux de Nicolas Roca Filella, détaillé dans le papier [Fil+22]. La caméra RGB-D séquence l'image contenant la gomme en utilisant une détection par couleur, utilise les fonctionnalité 3D pour déterminer les coordonnées de deux points de la bande de gomme. Utilisant l'équation de la chaînette, il est possible de calculer la tension dans l'objet manipulé, et d'en déduire une erreur de position. Ce code est une réutilisation de travaux déjà développé en python. Nous avons utilisé le script tel-quel, et mis en place une socket en utilisant la bibliothèque ZMQ pour remonter les données dans l'application. A noter que la limite des responsabilités respectives de `perception` et de `deformationControl` est floue : il aurait été correct de remonter l'erreur de tension du nœud `perception` et d'utiliser un module pour convertir l'erreur de tension en erreur de position dans le nœud `deformationControl`.

4.3.4 Nœud deformationControl

Pour simplifier le problème, et se concentrer sur la mise en place de la structure logicielle, le robot est uniquement commandé sur l'axe X. En utilisant l'erreur reçue du nœud `perception`, nous pouvons calculer une vitesse grâce à un régulateur PID. Ces termes proviennent des trois gains distincts appliqués. Le gain proportionnel s'applique directement sur l'erreur reçue. Le gain intégral s'applique sur une moyenne glissante des dernières erreurs, Le gain dérivé s'applique sur l'évolution de l'erreur, implémentable par simple soustraction de l'erreur précédente à l'erreur actuelle.

Le gain proportionnel permet d'effectuer l'asservissement. Trop faible, et le système sera trop lent. Trop élevé, le système sera instable. Les gains intégral et dérivé sont utilisés pour améliorer les performances du régulateur. Le gain Intégrale permet de stabiliser le système et d'éliminer des erreurs constantes. Trop élevé, il introduira des oscillations. La taille exacte de la moyenne glissante définie la durée gardée en mémoire pour le terme intégral, qui doit être adapté à la tâche. Le gain dérivé permet d'accélérer le régulateur, au risque d'une dégradation de la stabilité. Ce gain est très sensible aux bruits de mesure [ZN42]. En additionnant ces trois termes, nous obtenons une vitesse à appliquer à l'effecteur du robot, envoyé au nœud `skilledWorkcell`.

4.3.5 Nœud skilledWorkcell

Ce nœud reçoit les ordres provenant de `deformationControl`. Du côté du robot, l'interfaçage se fait grâce à l'Urcap (bloc proposé par le constructeur Universal Robot) externalControl, permettant de contrôler le robot avec des topics ROS. Du côté de l'ordinateur exécutant le logiciel SoftManBot, le contrôleur utilisé est une modification du pilote de base fourni par Universal Robot, permettant l'envoi de commandes exprimées en vitesses cartésiennes. Les commandes sont saturées avant l'envoi au robot, afin que ce dernier reste dans une fenêtre garantissant le bon fonctionnement de la perception.

4.3.6 Conclusion

Le but du travail sur ce cas de validation était de tester l'implémentation d'une solution à un problème simple en suivant le formalisme d'`Hades`. Ce cas de validation a permis de valider trois points :

- Tout d'abord, la praticité de l'architecture proposée : nous avons immédiatement su comment séparer le problème et où implémenter chaque fonction nécessaire à la tâche.
- Ensuite, la flexibilité : les développements existants au préalable ont été repris facilement pour être intégrés dans `Hades`.
- Enfin, la faisabilité : les sources génériques ont rempli leur rôle, le développement a uniquement impacté les sources spécifiques, comme ce qui été prévu.

4.4 Cas d'usage et d'intégration : banc de test académique

Cette section présente l'application d'`Hades` au banc de test académique, en rentrant dans le détail des modules développés à l'Institut Pascal et leur implémentation. La plupart de ces modules ont été développés dans le cadre du projet SoftManBot en temps que démonstrateurs indépendants, que nous avons restructuré, intégré et testé. Cette section détaille donc, en plus de nos propres développements, les développements de plusieurs partenaires.

La vue schématique de l'interfaçage avec le banc de test, dont les spécificités sont détaillées dans la section 1.2.2 est présentée dans la Fig. 4.9.

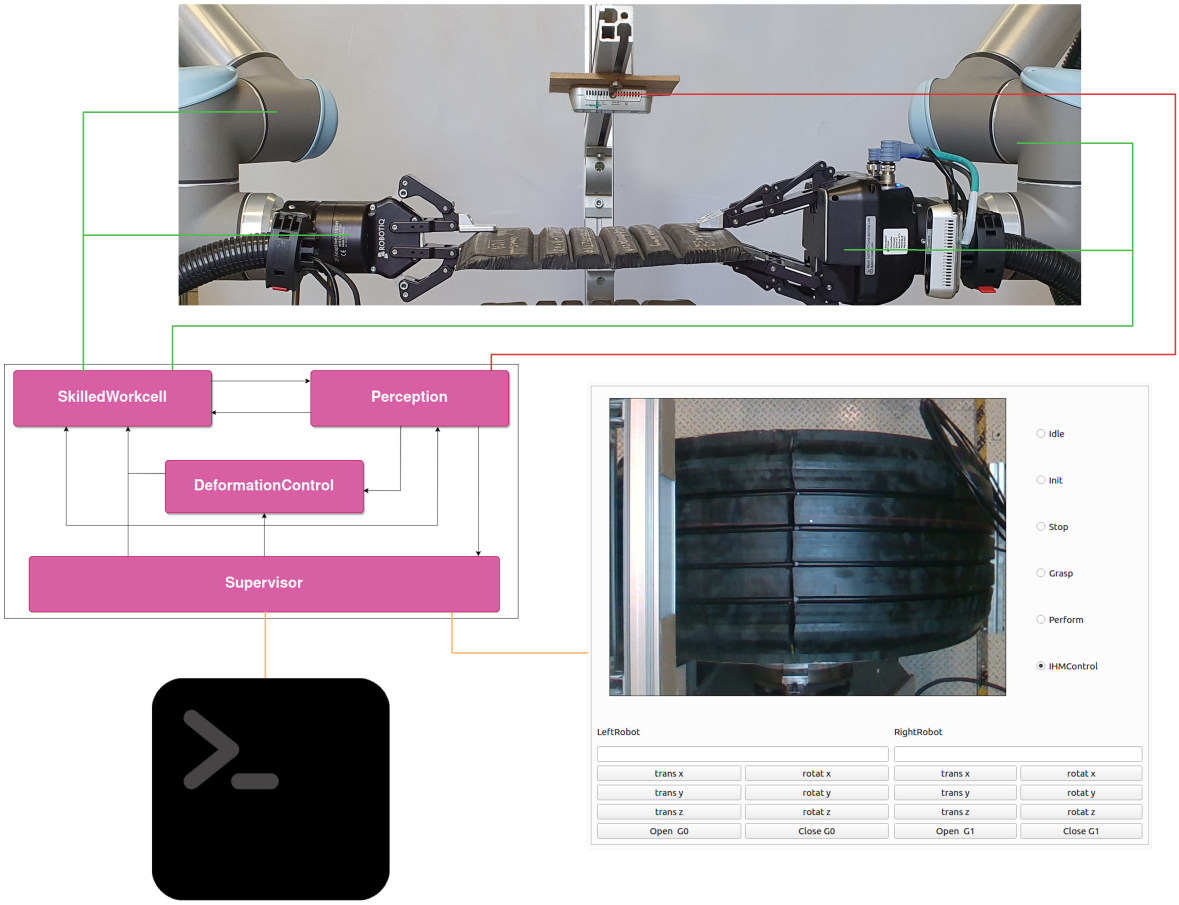

FIGURE 4.9: Configuration du banc de test académique

4.4.1 Modules de Supervisor

Interface Homme Machine

L'Interface Homme Machine implémentée est celle présentée Fig. 4.6. Fonctionnellement, nous l'avons implémentée d'abord comme une application pyqt –version python de l'outil de création d'interfaces graphiques Qt – autonome, avant de créer un package ROS à l'échelle du consortium. La communication de l'IHM avec le superviseur se fait par le biais de socket ZMQ, comme illustré dans les algorithmes 7 et 8.

Ces deux algorithmes fonctionnent de concert. L'algorithme 7 présente le pseudo-code python illustrant la mise en place des boutons de sélection d'état (ligne 1 à 3 et 5 à 7), ellipsé par soucis de concision et de clarté ligne 8. Les lignes 10 et 11 mettent en place un timer périodique permettant de cadencer la communication. La fonction définie ligne 13 est la Callback du click sur les boutons, sauvegardant l'état courant, sous forme d'entier de string contenant un entier et désactivant la radio. La fonction définie ligne 17 est la fonction de communication envoyant l'état courant à la machine d'état et récupérant l'information de fin de tâche. Une fois la tâche finie – condition ligne 20, la radio est réactivée, ligne 21.

Algorithme 7 Implémentation de la machine d'état piloté par l'IHM, côté IHM

```
1: radiobutton = QRadioButton("Init")
2: radiobutton.state = "init"
3: radiobutton.toggled.connect(self.radioOnClicked)
4:
5: radiobutton = QRadioButton("Idle")
6: radiobutton.state = "idle"
7: radiobutton.toggled.connect(self.radioOnClicked)
8: ...
9:
10: self.timer=QTimer()
11: self.timer.timeout.connect(self.sendState)
12:
13: def radioOnClicked() :
14: self._state = radioButton.state
15: self.radio.disable()
16:
17: def sendState() :
18: self._socket.pairSend(self._state.toInt())
19: msg = self._socket.pairReceive()
20: if msg.decode == stepDone then
21:     self.radio.enable()
22: end if
```

Algorithme 8 Implémentation de la machine d'état piloté par l'IHM côté superviseur

```
1: def getState() :
2: contents = s_recv(pairSocket)
3: if (isStepDone) then
4:     s_send(pairSocket, "StepDone")
5: else
6:     s_send(pairSocket, "StepNotDone")
7: end if
8: return std::stoi(contents)
```

L'algorithme 8 propose une implémentation de la fonction `getState`, utilisée par l'algorithme 6, donc séquençant le comportement entier du logiciel comme présenté précédemment. Il consiste en l'appel de socket pour récupérer l'état actuel de l'IHM (ligne 2), lui répondre en fonction du statut de la tâche (ligne 3 à 7), et enfin transformer la chaîne de caractère reçue en valeur numérique grâce à la fonction `std::stoi`.

4.4.2 Modules de SkilledWorkcell

contrôleur d'UR10 :

Les contrôleurs utilisés ont été développés par Laurent Lequievre, ingénieur à l'Institut Pascal, et Adrien Koessler, Post-doctorant à l'Institut Pascal. Nous proposons ici une encapsulation standardisée du logiciel gérant les robots, qui a pour but d'abstraire pour l'utilisateur les robots et

```hpp
#ifndef UR10_HPP
#define UR10_HPP
#include "../common/genericLogic.hpp"

typedef enum
{
  ARM1,
  ARM2
}currentRobotArm;
typedef enum
{
  NO_CONTROLLER,
  SPEED_CONTROLLER,
  POSITION_CONTROLLER,
  JOINT_CONTROLLER
}currentRobotController;

void UR10RosInit(void);
void stopCurrentController(currentRobotArm armToStop);
void startCurrentController(currentRobotArm armToStart, currentRobotController controllerToStart);
currentRobotController getCurrentController(void);
void jointMoveArm(currentRobotArm armToMove, const std::vector<float> &jointOrder_rad);
void cartesianPositionMoveArm(currentRobotArm armToMove, pose armTargetPose);
void cartesianVelocityMoveArm(currentRobotArm armToMove, pose armSpeedOrder);
pose getArm1Pose(void); //Return relative pose of the arm
pose getArm2Pose(void); //Return relative pose of the arm

#endif //UR10_HPP
```

FIGURE 4.10: Header des robots

contrôleurs utilisés, afin d'axer le raisonnement sur "que dois faire le robot" plutôt que sur "comment faire bouger le robot". Cette abstraction est représentée dans le header présenté Fig. 4.10.

Les contrôleurs utilisent KDL. Nous utilisons un contrôleur articulaire pour les tâches `Init` et `Stop`, afin de maîtriser la configuration des robots. En choisissant avec soin les positions des robots, cela nous permet de travailler dans un environnement sans singularité. Nous utilisons un contrôleur en position cartésienne pour la saisie de la gomme, un contrôleur en vitesse cartésienne pour effectuer la tâche de soudure. Les contrôleurs interagissent avec le robot grâce à l'URCAP "ExternalControl".

Préhenseurs ROBOTIQ

La configuration matérielle du banc de test nous a imposé l'usage de deux préhenseurs différents : un bras était équipé d'un préhenseur Robotiq 2F, l'autre d'un préhenseur Robotiq 3F. L'intégration de ces préhenseurs a été simple grâce aux packages ROS fournis par le fabricant. Nous avons cependant figé une interface unique, présenté dans la Fig. 4.11. L'intérêt est d'abstraire complétement les détails d'implémentations aux utilisateurs des pinces, qui sont disponibles en tant que service défini dans le fichier specificSupervisor.cpp. Le service appelle les fonctions `gripper_open` et `gripper_close`, n'étant pas impacté par les changements matériels.

4.4.3 Modules de Perception

Drivers de caméra

Le header de camera, présenté Fig. 4.12 a été utilisée pour encapsuler la caméra monoculaire embarquée dans un ordinateur portable avec les mêmes prototypes de fonctions que la caméra In-

```hpp
#ifndef GRIPPER_HPP
#define GRIPPER_HPP

void gripper_init(int gripperNumber);
void gripper_open(int gripperNumber);
void gripper_close(int gripperNumber);

#endif //GRIPPER_HPP
```

FIGURE 4.11: Header des grippers

```hpp
#ifndef REALSENSE_HPP
#define REALSENSE_HPP

#include <opencv2/opencv.hpp>
#include "../../common/genericLogic.hpp"
#include <librealsense2/rs.hpp>     //Necessary for camera_getDepthOrig

//Prototype of camera hpp
void camera_RosInit(void);
void camera_init(void);
void camera_update(void);//Call cyclically. Create new frames once per call.
cv::Mat camera_getFrame(void);
cv::Mat camera_getDistCoef(void);
cv::Mat camera_getDepth(void);//Throw in monocular cams
cv::Mat camera_getCalibrationMatrix(void);
std::vector<float> camera_getCalibrationMatrixVector(void);
pose camera_2D_to_3D(float pixel[2]); //Turning a pixel into 3D point


rs2::depth_frame camera_getDepthOrig(void);
rs2_intrinsics camera_getIntri(void);

#endif //REALSENSE_HPP
```

FIGURE 4.12: Header de camera

tel Realsense D435 utilisée dans les bancs de test. Bien sûr, il faut que la méthode de perception ne s'appuie pas sur les fonctions de déprojection, qui lancent des exceptions pour les drivers du hardware non compatible. L'intérêt du header vient des fonctions `getFrame` et `getDepth`. Appelées une fois par boucle, cela permet à plusieurs algorithmes d'exploiter la même frame sans la consommer.

Mise en œuvre de la méthode de perception ROBUSfT

La méthode de perception ROBUSfT [SB+23] est une variante de Shape-From-Template permettant de reconstruire un objet surfacique en 3D. L'algorithme nécessite un patron visuel de l'objet à suivre, nommé "Template". Idéalement, le patron doit être fortement texturé. Une "forte texture" permet de connaître la position dans le patron d'un petit élément. Cette méthode nécessite de percevoir l'intégralité du template, ce qui exclue les déformations importantes de l'objet. Le résultat du maillage appliqué à l'objet peut être vu dans la Fig. 3.7.

Intégration dans `Hades` : Cette méthode a été développé par Mohammadreza Shetab-Bushehri,

```
1  #ifndef ROBUSFT_CPU
2  #define ROBUSFT_CPU
3
4  #include "../../common/mesh.hpp"
5  #include "../../common/genericLogic.hpp"
6  #include <opencv2/opencv.hpp>
7
8  void robusft_cpuRosInit(void);
9  void robusft_Init(void);
10
11 softmanbot::mesh *robusft_update(void);
12
13 cv::Mat robusft_getFrame(void);
14 void setHandledNodes(pose endEffectorPositionLeft, pose endEffectorPositionRight);
15 std::vector<int> getHandledNodesLeft(void);
16 std::vector<int> getHandledNodesRight(void);
17
18 #endif // ROBUSFT_CPU
```

FIGURE 4.13: Header de la perception ROBUSfT

doctorant à l'Institut Pascal. Ma contribution vient du travail de restructuration pour intégrer sous forme de modules interchangeables les développements, les adapter et les tester de manière approfondie sur le cas d'usage.

L'interface avec le reste de l'architecture se fait via le header présenté dans la Fig. 4.13.

Outre l'ajout de fonctions d'initialisation (ligne 8 avant toute utilisation, ligne 9 pour remettre dans un état de départ maîtrisé), nous avons rajouté des fonctions utilitaires permettant de connaître quels sont les nœuds les plus proches des points de préhensions (fonctions `setHandledNodes` et `getHandledNodes`). Cette méthode de perception fonctionne en appelant à chaque itération la fonction `robusft_update` pour obtenir le maillage, et la fonction `robusft_getFrame` pour obtenir et montrer la frame augmentée du maillage.

Synthèse du test de la méthode ROBUSfT : Les forces de la méthode sont sa vitesse impressionnante, sa robustesse et sa précision.

- La vitesse, parce que la méthode fonctionne à une vitesse de 15Hz en 480p malgré peu d'optimisations
- La robustesse, de deux manières. Premièrement, la méthode fonctionne avec de la détection et non pas du suivi. Le réseau de points détecté à l'instant $k+1$ ne dépend pas de celui détecté à l'instant k, ce qui implique que l'objet à détecter peut sortir du champ de la caméra d'un côté, ré-entrer de l'autre, et être détecté dès la première image où le patron est suffisamment visible. Deuxièmement, la méthode implémente dans ses rouages internes une phase de suppression des valeurs aberrantes. Cette phase, contribution scientifique de la méthode vis-à-vis des autres variantes de Shape-From-Template, améliore grandement l'efficacité. Les points détectés hors du cluster principal sont supprimés et estimés, permettant d'améliorer la détection du patron.
- La précision, parce que l'algorithme semble précis au pixel près. Cela nous a permis d'obtenir des alignements visuellement parfaits - mesurés à moins d'un demi millimètre.

Mise en œuvre de la méthode de perception suivi de forme LARAP

LARAP signifie Latticed As-Rigid-As-Possible. Le principe de la méthode est d'englober le nuage de point représentant l'objet déformable dans une boîte – appelée Lattice, que nous tra-

duirons en treillis – et de suivre l'évolution de la forme de l'objet avec le modèle ARAP. Cette méthode nécessite également un patron, celui du nuage de point non déformé à suivre et percevoir. La méthode extrait le nuage de point à proximité du treillis, calcule les vecteurs orthogonaux à la surface, et les fait correspondre avec le patron connu pour actualiser la position et la déformation actuelle de l'objet. Pour ce qui est des déformations elles-mêmes, c'est le treillis qui est déformé avec le modèle ARAP pour faire correspondre le patron déformé à la réalité perçue. La forme du treillis déformé permet donc de déduire la forme de l'objet déformé.

Cette méthode a été développé par Mohammadreza Shetab-Bushehri, doctorant à l'Institut Pascal. Ma contribution vient du travail de restructuration pour intégrer sous forme de modules interchangeables les développements, les adapter et les tester de manière approfondie sur le cas d'usage.

```
1  #ifndef LARAP_SHAPE_TRACKING
2  #define LARAP_SHAPE_TRACKING
3
4  #include "../../common/mesh.hpp"
5  #include "../../common/genericLogic.hpp"
6  #include <opencv2/opencv.hpp>
7
8  void larapShapeTracking_RosInit(void);
9  void larapShapeTracking_Init(void);
10 softmanbot::mesh  larapShapeTracking_update(void);
11 void larapShapeTracking_SetEndEffectorPositions(pose endEffectorLeftPose, pose endEffectorRightPose);
12 cv::Mat larapShapeTracking_getFrame(void);
13 void larapShapeTracking_onGrasp(pose endEffectorLeftPose, pose endEffectorRightPose);
14 void larapShapeTracking_startTracking(void);
15 softmanbot::mesh larapShapeTracking_trackFixedRubber(void);
16
17 #endif //LARAP_SHAPE_TRACKING
```

FIGURE 4.14: Header de la perception LARAP

Intégration dans `Hades` : Le header utilisé pour interfacer le module avec le nœud perception est montré dans la Fig. 4.14. Le header est très similaire à celui de la perception ROBUSfT, présenté dans la Fig. 4.13, permettant de facilement passer d'une perception à l'autre. Les fonctions fonctionnent de manière similaire à ce qui a été présenté pour la méthode ROBUSfT. Par exemple, ligne 8 et 9, les fonctions d'initialisation `larapShapeTracking_RosInit` et `larapShapeTracking_Init`. La fonction `larapShapeTracking_RosInit` est appelée avant la machine d'état pour appeler les constructeurs et les premières initialisation, et la fonction `larapShapeTracking_Init`, qui est appelée à chaque tâche `Init`, pour réinitialiser des variables le nécessitant comme la position de base du patron à suivre ou l'état des filtres de post-traitement. Les différences avec l'interface de ROBUSfT viennent des fonctions `larapShapeTracking_onGrasp` - effectuant le code de perception nécessaire lors de la fermeture des doigts, la fonction `larapShapeTracking_startTracking`, permettant de déclencher le suivi de forme, et une fonction dédiée à la détection de la partie fixe de gomme, `larapShapeTracking_trackFixedRubber`

Synthèse du test de la méthode ROBUSfT : Les performances sont à la limite du tolérable pour le cas d'usage, est plus lente et peut souffrir de décrochage si les mouvements sont trop rapide : la méthode suit la forme au lieu de la détecter. Cependant, la méthode propose une solution au problème de détection non-texturée de la gomme.

4.4.4 Modules de DeformationControl

Nous pouvons définir deux sous-catégories au problème de stratégie : la catégorie dans laquelle nous nous concentrons uniquement sur les mouvements effectués par le robot, regroupant les méthodes d'apprentissage par démonstration, et la catégorie dans laquelle nous nous concentrons davantage sur la forme de l'objet et où les robots sont un moyen d'influencer cette dernière. La première catégorie sera appelée "solutions en boucle ouverte" et la deuxième "solutions en boucle fermée". Au sein de la deuxième catégorie, un type de stratégie particulièrement adapté au problème à traiter est le contrôle par retour d'état : dans notre cas, on s'intéresse à l'état de l'objet manipulé, représenté par les coordonnées des nœuds formant le maillage de l'objet.

Les stratégies en boucle fermée sont plus difficile à mettre en œuvre car elles comportent davantage de dépendances : elles nécessitent une perception de l'objet déformable, une modélisation et un contrôle en boucle fermée, alors que les stratégies relevant de la premières catégorie ne nécessitent qu'un contrôle en boucle ouverte.

Cette section présente l'implémentation de stratégies provenant de ces deux catégories.

Stratégie en boucle ouverte : Dual-Quaternion based Dynamic Movement Primitives

Ces travaux ont été publiés à ICRA 2023 [Cha+23]. Cette stratégie a été développée par Rohit Chandra, post-doctorant de l'institut Pascal, premier auteur et moi-même, second auteur.

FIGURE 4.15: Objet déformable utilisé pour les expériences

Nous avons conduit les expériences sur le banc de test académique, remplaçant la gomme par une barre en mousse représentée Fig. 4.15. L'objet a été texturé, ce qui nous a permis de suivre l'évolution de la déformation aussi bien dans les phases d'apprentissage que le phases où nous avons rejoué les trajectoires apprises. Cette déformation est mesurée grâce au module ROBUSfT. Les marqueurs ArUco ont permis d'évaluer dans un premier temps les rotations et translations imposées aux points de contrôle, pour étudier la faisabilité pratique de la méthode. Pour améliorer la précision du suivi, nous avons utilisé le système de capture de mouvement "Qualisys" dans nos expériences finales.

Intégration dans `Hades` : Nous avons simplement généré les trajectoires en vitesse cartésiennes, que nous avons envoyé aux robots. L'adaptation de la trajectoire aux conditions initiales ou finales est effectuée avec l'utilisation de fonctions classiques de transformation dans l'espace, grâce à la bibliothèque logicielle tf2 de ROS.

Synthèse du test de la stratégie Dual-Quaternion based Dynamic Movement Primitives :

Le principe de la stratégie : enregistrer une séquence de trajectoire appliquée à un objet déformable pour effectuer une tâche. Le formalisme utilisé est nommé Dual-Quaternion based Dynamic Movement Primitives. L'usage de Dual-Quaternion permet de coupler les rotations et translation en un seul mouvement de vis. La trajectoire apprise est interpolée aux position de départ et position désirée en ligne, permettant d'adapter la stratégie de learning : on ne rejoue simplement pas une trajectoire apprise, on adapte la trajectoire aux conditions initiales et finales désirées. La trajectoire est apprise en interpolant les mouvement de vis à une collection de Gaussiennes pondérées. L'interpolation est effectuée avec l'algorithme ScLERP [Kav+06b], et permet de créer une trajectoire continue. De plus, l'interpolation joue le rôle de filtre, lissant les erreurs de perception et les éventuels discontinuités ou tremblement de l'opération humaine.

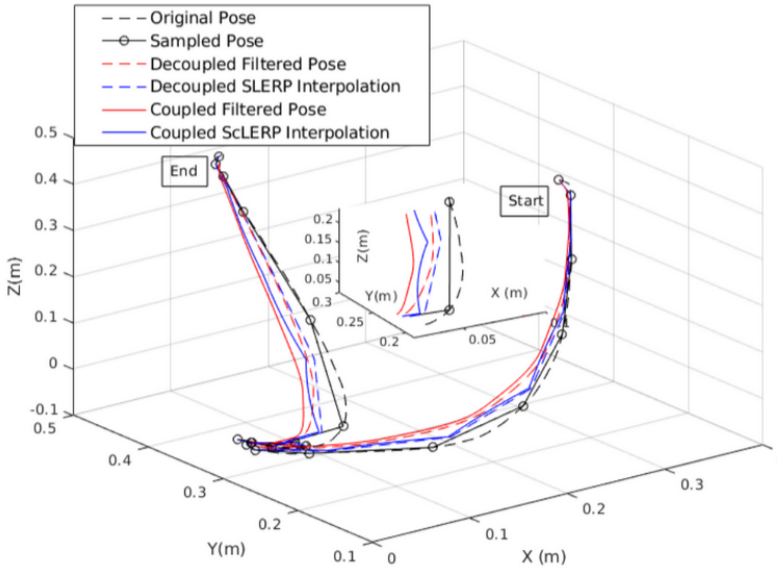

FIGURE 4.16: Résultat de la stratégie appliquée à un seul manipulateur pour une tâche de Pick and Place

Extrait du papier [Cha+23]. Figure réalisée par le premier auteur du papier. La figure présente les diverses trajectoires, avec les différents filtres appliqués.

La formulation sous forme de Dual-Quaternion présente l'avantage de coupler les translations et rotations de l'opérateur humain. Prendre en compte ce couplage permet d'obtenir un meilleur alignement pour la trajectoire résultante, montrée dans la Fig. 4.16.

Approche de modèle : As-Rigid-As-Possible

Ce modèle a été implémenté dans sa formulation surfacique, adapté à un objet planaire, par Miguel Aranda et Mohammadreza Shetab-Bushehri [SB+22]. Ma contribution a été de reprendre leur code suivant l'antipattern du tas de boue comprenant le modèle, la logique des robots, la perception et la logique haut-niveau, puis de le séparer en modules.

Intégration dans `Hades` : Les interfaces sont présentées dans la Fig. 4.17. Le modèle doit être initialisé (fonction `ARAP_init`), et le patron utilisé doit être explicité (avec la fonction `ARAP_setMeshFromTemplate`), ainsi que les nœuds du réseau directement manipulés par les robots (fonction `ARAP_setRobotNodes`). La Jacobienne est formattée comme une matrice de la bibliothèque Eigen (fonction `ARAP_jacobian`). Avant de calculer cette jacobienne, il faut bien sûr actualiser le maillage courant de l'objet (fonction `ARAP_setMeshFromPerception`). On peut noter la cohérence avec les fonctions des modules de perecption, en particulier pour les types de retour.

Synthèse du test de l'approche ARAP : Dans ce paradigme, l'objet est considéré comme déformable, mais de mauvaise volonté. Il suit les mouvements du robots en se déformant aussi peu que possible, d'où le nom As-Rigid-As-Possible. La jacobienne est calculée en simulant des mouvement unitaires sur tous les degrées de liberté des robots manipulateurs. Ce modèle calcule la forme simulée avec des critères géométriques plutôt que physiques. Cela permet de soulager la charge de calcul, tout en restant suffisamment précis pour notre cas d'usage.

```hpp
#ifndef ARAP_JACOBIAN_HPP
#define ARAP_JACOBIAN_HPP

#include "../../common/genericLogic.hpp"
#include "../../common/mesh.hpp"
#include "std_msgs/Float64MultiArray.h"
#include <Eigen/Dense>

void ARAP_init(void);
void ARAP_setMeshTemplate(const std_msgs::Float64MultiArray& msg);
void ARAP_setMeshFromPerception(mesh currentShape);
void ARAP_setRobotNodes(std::vector<int> leftRobotNodes, std::vector<int> rightRobotNodes);

Eigen::MatrixXd ARAP_jacobian(mesh ObjectState);//The jacobian links movement to rigid set position
//To get the correct and relevant out, you need to call the following function on the vector out of your controller
std::vector<pose> rigidSetToCamera(Eigen::VectorXd vel_p);

#endif
```

FIGURE 4.17: Interface de l'implémentation du module de modèle ARAP

Stratégie d'asservissement de forme : Proportional shape-servoing Control

Ce contrôleur proportionnel a été implémenté par Miguel Aranda et Mohammadreza Shetab-Bushehri [SB+22].

```hpp
#ifndef CONTROLLER_PROPORTIONAL_HPP
#define CONTROLLER_PROPORTIONAL_HPP

#include "../../common/mesh.hpp"
#include <Eigen/Dense>

void Controller_init(void);
void setControllerDesiredShape(softmanbot::mesh targetShape);
void setControllerCurrentShape(softmanbot::mesh currentShape);

Eigen::VectorXd controller(Eigen::MatrixXd jaco);

#endif
```

FIGURE 4.18: Interface de l'implémentation du contrôleur jacobien-proportionnel

Intégration dans `Hades` : Le contrôleur a été implémenté avec une interface le réinitialisant (fonction `Controller_init`), deux setters pour la forme désirée et la forme actuelle (fonctions `setControllerDesiredShape` et `setControllerCurrentShape`), et une fonction `controller`, prenant en paramètre la jacobienne remontée par le modèle, calculant les commandes à envoyer aux robots sous forme de vecteur Eigen empilé.

L'intérêt de cette séparation du contrôleur et du modèle est la possibilité de changer l'un ou l'autre de manière indépendante et minimale, permettant des comparaisons plus justes entre les différentes approches et stratégies.

Synthèse du test de la stratégie proportionnelle : Cette stratégie de contrôle calcule l'erreur de forme, vecteur composé de la différence nœud à nœud en x, y, z entre la forme détectée par la perception et la forme souhaitée de l'objet. En démarrant de l'équation d'état du système

$$x(k+1) = x(k) + J(k)\,u(k)\,dt \tag{4.1}$$

Avec $k \in \mathbb{N}$ le temps discrétisé, $dt \in \mathbb{R}$ le pas de temps, $x \in \mathbb{R}^{3n}$ le vecteur d'état formé par les coordonnées empilées des nœuds, et n le nombre de nœuds dans le réseau de l'objet. $u \in \mathbb{R}^{6m}$

représente la commande envoyée au robot sous forme de vecteurs de vélocités, avec m le nombre de robots. $J \in \mathbb{R}^{3n \times 6m}$ est la Jacobienne de déformation.

$$x = \begin{pmatrix} x_1 \\ y_1 \\ z_1 \\ x_2 \\ y_2 \\ z_2 \\ \vdots \end{pmatrix}_{3n \times 1} \quad \text{and} \quad u = \begin{pmatrix} v_{x_1} \\ v_{y_1} \\ v_{z_1} \\ w_{x_1} \\ w_{y_1} \\ w_{z_1} \\ \vdots \end{pmatrix}_{6m \times 1} \tag{4.2}$$

Nous définissons le vecteur d'erreur de forme $e \in \mathbb{R}^{3n}$ de la manière suivante :

$$e(k) = x(k) - x_d(k) \tag{4.3}$$

Nous explicitons au lecteur que cette formulation est la même que pour Optimal Shape Servoing, décrite dans la section 3.2.

Avec cette formulation, la manière la plus simple de commander les robots est le retour d'état proportionnel, formulé

$$u(k) = -K J^{+} e(k) \tag{4.4}$$

avec $K(k) \in \mathbb{R}$ le gain proportionnel de la commande, et J^{+} la pseudo-inverse au sens de Moore-Penrose de la Jacobienne J.

Contextualisation des résultats : L'association de perception, contrôle et modèle proposée par Mohammadreza Shetab-Bushehri et Miguel Aranda a permis la validation de plusieurs points. Premièrement, le problème consistant à atteindre une forme connue atteignable semble solvable à la fois en simulation et dans le monde réel. Une forme est considérée atteignable si nous pouvons l'observer sous une configuration de saisie donnée. Cependant, nous n'avons pas de preuve formelle pour ce constat. Deuxièmement, notre problème non linéaire volumique peut être réduit à un problème surfacique, solvable avec une stratégie proportionnelle avec linéarisation de la Jacobienne recalculée à chaque pas de temps.

Stratégie d'asservissement de forme : Optimal Shape Servoing

```
1  #ifndef CONTROLLER_OPTIMAL_HPP
2  #define CONTROLLER_OPTIMAL_HPP
3
4  #include "../../common/mesh.hpp"
5  #include <Eigen/Dense>
6
7  void Controller_init(void);
8  void setControllerDesiredShape(softmanbot::mesh targetShape);
9  void setControllerCurrentShape(softmanbot::mesh currentShape);
10
11 Eigen::VectorXd controller(Eigen::MatrixXd jaco);
12
13 #endif
```

FIGURE 4.19: Interface de l'implémentation du contrôleur Optimal Shape Servoing

Cette stratégie d'asservissement de forme est détaillée dans le chapitre 3. Son interface est présentée dans la Fig. 4.19. On peut noter l'exacte similarité avec l'interface du contrôleur proportionnel, présenté dans la Fig. 4.18. Le fait que ces interfaces soient identiques valide la réutilisabilité : il est possible de modifier les sources existantes pour améliorer une fonctionnalité à échelle modulaire, sans se poser des questions d'architecture globale.

4.5 Cas d'usage et d'intégration : banc de test industriel

Cette section présente l'application d'Hades au banc de test industriel, en reprenant les modules développés lors de la section précédente, les applicant au cas d'usage, et développant des solutions alternatives en cas d'inadéquation au problème à résoudre.

La vue schématique de l'interfaçage avec le banc de test, dont les spécificités sont détaillées dans la section 1.2.1, est présentée dans la Fig. 4.20.

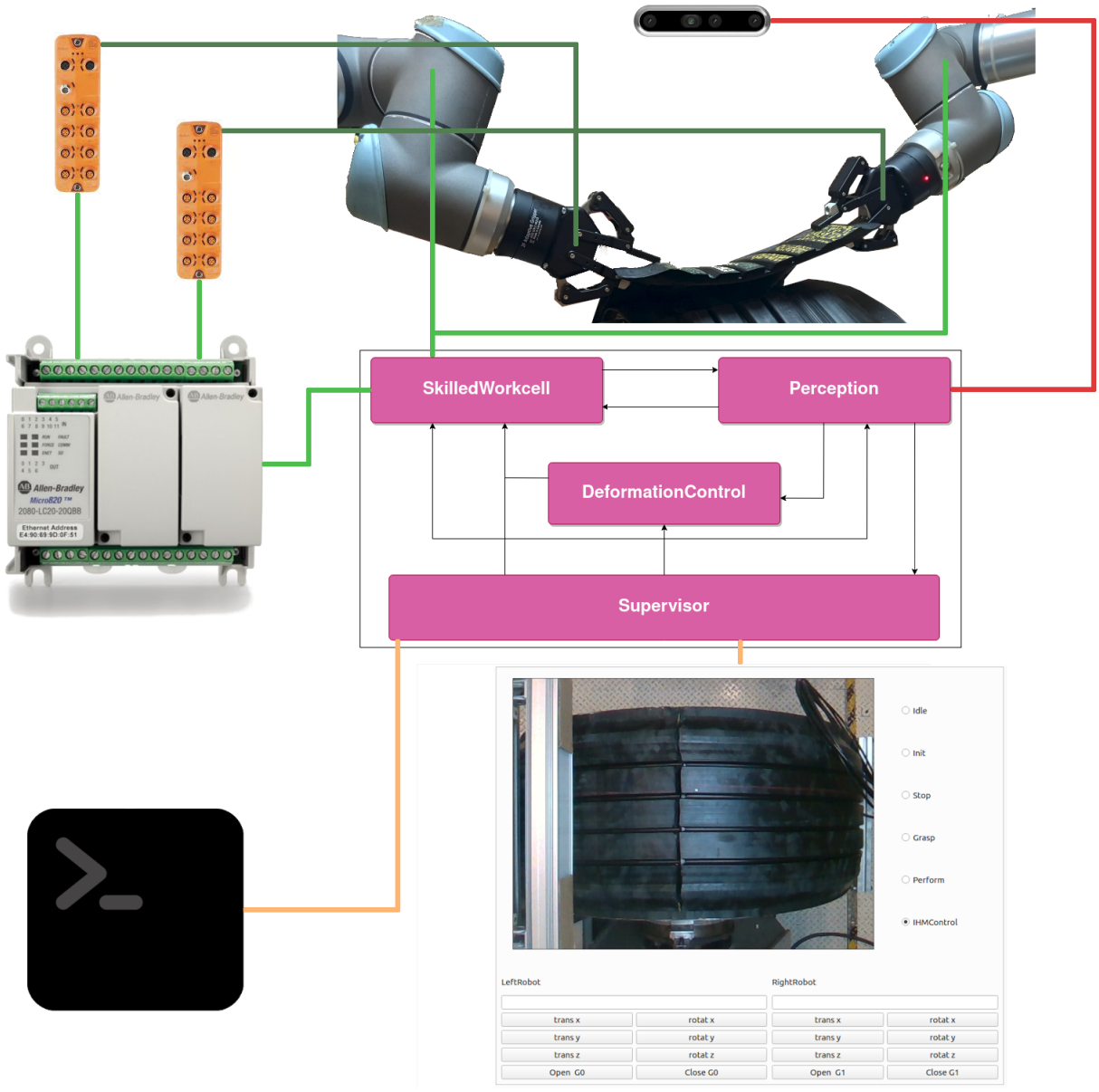

FIGURE 4.20: Configuration du banc de test industriel

4.5.1 Modules de Supervisor

Interface Homme Machine

Nous avons repris le module développé pour le banc de test académique. Le module a été porté facilement et rapidement, sans conflit ou problème majeur.

L'intérêt principal de l'usage de l'IHM est le prototypage et la praticité de pouvoir intégrer le retour visuel augmenté, des déplacements de robots ainsi que des actions de préhenseur de manière interactive.

Machine d'état piloté par PLC

Pour se rapprocher de l'industrialisation, nous avons implémenté un pilotage par le PLC du banc de test industriel. Le superviseur agit ainsi comme une boîte à lettre, recevant les consignes de haut niveau du PLC, comme montré dans l'algorithme 9. Cet algorithme présente une implémentation en pseudo-code C++ de la fonction `getState`, utilisée par l'algorithme 6, donc séquençant le comportement entier du logiciel comme présenté précédemment. De plus, le superviseur doit remonter l'état des tâches accomplies au PLC, comme montré dans l'algorithme 10 - on peut faire l'analogie avec les boutons radio de l'IHM, nécessitant de savoir si la tâche en cours était terminé, comme discuté dans l'explication de l'algorithme 7.

Algorithme 9 Implémentation de la machine d'état piloté par le PLC

```
1: softmanbot::state getState(){
2: return clientPlcRead.call(currentTaskTag)
3: }
```

Algorithme 10 Exemple de tâche pilotée par PLC : Init

```
1: if isSkilledWorkcellDone && isPerceptionDone then
2:    clientPlcWrite.call(statusStepTag, JOBS_DONE)
3: end if
```

L'intérêt principal de l'usage du PLC est la sécurité : en plus du hardware, les états sont complètement bloqués en cas d'arrêt d'urgence, et il est même possible de déclencher des arrêts d'urgence logiciels. Cela permet d'être conforme aux règles de sécurité en vigueur pour les machines destinées aux lignes de production.

4.5.2 Modules de SkilledWorkcell

Communication avec le PLC

De par les contraintes du banc de test industriel, nous devons communiquer avec le PLC. Pour ce faire, nous avons utilisé la librairie Python Pylogix, déjà utilisé par les ingénieurs Michelin Nicolas Bard et Nicolas Roca-Filella. Pour que la communication fonctionne, il faut évidemment que le logiciel côté PLC soit écrit pour traiter et répondre aux messages avec un protocole de communication spécifié en avance.

Nous avons donc encapsulé les fonctions nécessaires python dans un fichier `plcComm.py`, interfacé par le Header représenté Fig. 4.21. Les fonctions sont ensuite appelées grâce à l'embedding, en incluant le header `<Python.h>`. Il faut être prudent avec deux points : la durée en scope des objets et le multithreading. Les modules importés seront déchargés en sortant du scope.

```
#ifndef PLC_HPP
#define PLC_HPP

#include <string>

const std::string softmanbotStatusMasterTag = "PLC_REQUEST";
const std::string softmanbotStatusValidationTag = "SMB_GOT_REQUEST";
const std::string softmanbotStatusStepTag = "SMB_STEP";
const std::string watchdogMasterTag = "PLC_WATCHDOG";
const std::string watchdogValidateTag = "SMB_WATCHDOG";
typedef enum
{
  STATUS_REQUESTED,
  WORK_IN_PROGRESS,
  JOBS_DONE
}StepStatus;

void PLC_Request_Init(void);
void PLC_Request_WriteTag(const std::string &tagName, int valueToWrite);
int  PLC_Request_ReadTag(const std::string &tagName);
void PLC_Request_End(void);

#endif //PLC_HPP
```

FIGURE 4.21: Header de la communication avec le PLC

Les initialisations faisant appel à l'embedding doivent donc garder les modules en mémoire pour toute la durée où les fonctions seront potentiellements appelées. Le code python appelé passera par le Global Interpreter Lock, autorisant uniquement un seul thread à avoir accès à Python dans un même process. Dans les faits, cela induit des bugs lorsque plusieurs modules sont embarqués dans des fichiers différents - bugs que nous avons résolu en effectuant tous les traitements d'interfaçage Python dans le même fichier dans la solution finale, au dépend de la clareté et la lisibilité.

Contrôleur d'UR10 compatible ROS

Malgré un transfert et une installation facile, nous nous sommes rendus compte de limitations du contrôleur utilisé dans le cas d'usage réel, par rapport au cas simulé en laboratoire. La paramétrisation de la cellule déclenche le mode limité des robots dès qu'un utilisateur est à proximité. Le contrôleur et le module externalControl fonctionne mal dans ce cas : les trajectoires générées par le contrôleur cartésien atteignent les positions avec du dépassement, le contrôleur en vitesse ne bouge pas à la vitesse souhaitée mais proportionnellement au facteur de limite du robot. Enfin, la gomme manipulée est beaucoup plus lourde. Les tests du banc de test ont été effectués avec un morceau de gomme formant un carré de 20 cm de côté. La gomme fait le tour complet du cylindre dans le cas d'usage réel, d'où le couple robot nécessaire plus élevé .

contrôleur UR10 compliant

Nous avons donc repris un autre contrôleur développé par un membre du consortium, Carlos Vélez García, ingénieur à Insescop. Le contrôleur fonctionne avec le protocole modbus TCP, en envoyant directement des ordres appliqués par le robot et ses propres contrôleurs interne. Ce driver est originellement en Python - nous avons embarqué le code Python pour qu'il soit appelé par

l'exécutable C++ de la même manière que pour l'interfaçage avec le PLC. La bibliothèque fournie comportait un contrôleur compliant en position cartésienne ainsi qu'un contrôleur articulaire. J'ai développé un contrôleur compliant en vitesse cartésienne pour les besoins du projet. Le header utilisé est strictement identique à celui présenté dans la Fig. 4.10, validant les choix architecturaux de l'interface avec les robots.

Préhenseurs Zimmer

Les préhenseurs utilisés pour la solution finale sont ceux développés par le partenaire allemand Zimmer. Nous avons testé et intégré de nombreux préhenseurs - 3 séries de préhenseurs électriques, une série de préhenseurs pneumatiques - toujours utilisant l'interface de la Fig. 4.11. Les grippers utilisent un procédé de communication propre à chaque modèle, piloté par IOLink. Nous avons utilisé le PLC comme une boîte à lettre pour s'interfacer avec les modules IOLink.

4.5.3 Modules de Perception

Méthode de perception : ROBUSfT

Intégration dans `Hades` : Le module a été porté aisément d'un banc de test à l'autre, avec des résultats cohérents sur le banc de test industriel avec ce qui a été développé du côté académique. Nous avons bien sûr dû rajouter de la texture à la gomme, comme montré dans la Fig. 4.22.

Synthèse du test sur le cas d'usage réel : Nous savions dès le départ que cette méthode ne serait pas adaptée au cas d'usage finale : il n'est pas envisageable de systématiquement texturer les bandes de roulement pour les besoins de la perception. Cependant, cette méthode nous a permis de créer une boucle fermée, comprenant une interface logicielle avec les robots, les préhenseurs, la caméra, et surtout le contrôle. Le logiciel résultant de l'association de ces modules a permis de tester des hypothèses de faisabilité dans le monde réel et de tester les premières approximations non préjudiciables à l'accomplissement de l'objectif final.

Cette méthode de perception reste relativement facile à mettre en œuvre et peu contraignante, et est parfaitement adaptée pour des travaux de recherche ou de prototypage industriel.

FIGURE 4.22: Tread texturé pour les tests de ROBUSfT avec controlleur OSS sur le banc de test Michelin

Méthode de perception : suivi de forme LARAP

Intégration dans Hades : Le portage du module d'un banc de test à l'autre a été aisé - pour ce qui est de l'aspect purement logiciel. Cependant, la mise au point a nécessité des efforts démesurés pour obtenir les résultats détaillés ci-dessous, alors que la méthode a été expressément développée avec le cas d'utilisation à l'esprit.

Synthèse du test sur le cas d'usage réel : La méthode a présenté de nombreuses limitations et problème quand confrontée au cas d'usage réel :

1. Premier problème : lorsque les deux parties de la gomme sont proches, les nuages de points glissent en se superposant en laissant la soudure au milieu des treillis. La perception devient alors inutile. Nous avons résolu ce problème grâce à des filtres 3D intelligents : détection de la partie fixe à la fin de la tâche Init, calcul des points cibles à partir de celle-ci, pré-filtrage des points pour que la perception ne décroche plus.
2. Second problème : le manque de précision de la méthode. La précision du suivi de nuage de points est mauvaise et soumise à un bruit de plus ou moins trois millimètres. Cette précision est au delà des attentes pour la tâche.
3. Troisième problème : le suivi d'une partie d'un objet. À l'Institut Pascal, nous avons suivi l'ensemble de l'objet en cours de manipulation, un carré de gomme mesurant trente centimètres sur trente centimètres. Dans le cas plus réaliste, la bande de gomme est continue et fait le tour complet du pneu. La perception est incapable de suivre, avec un "glissement" de

la zone perçue le long du caoutchouc, ce qui accroît davantage les imprécisions et l'impraticabilité, comme le montre Fig. 4.23. Nous avons résolu ce problème en utilisant la position de l'effecteur, en supposant que nous saisissons toujours au même point. Cela implique que la méthode nécessite une autre entrée pour effectuer une saisie appropriée, et dépend de la précision de la saisie. La stratégie de saisie utilisée est présentée dans l'annexe B.

En conclusion, cette méthode semble trop limitée pour le cas d'usage. De plus, la maturité acquise sur le problème indiquerai qu'il n'y ait pas besoin d'autant de points suivis pour accomplir la tâche.

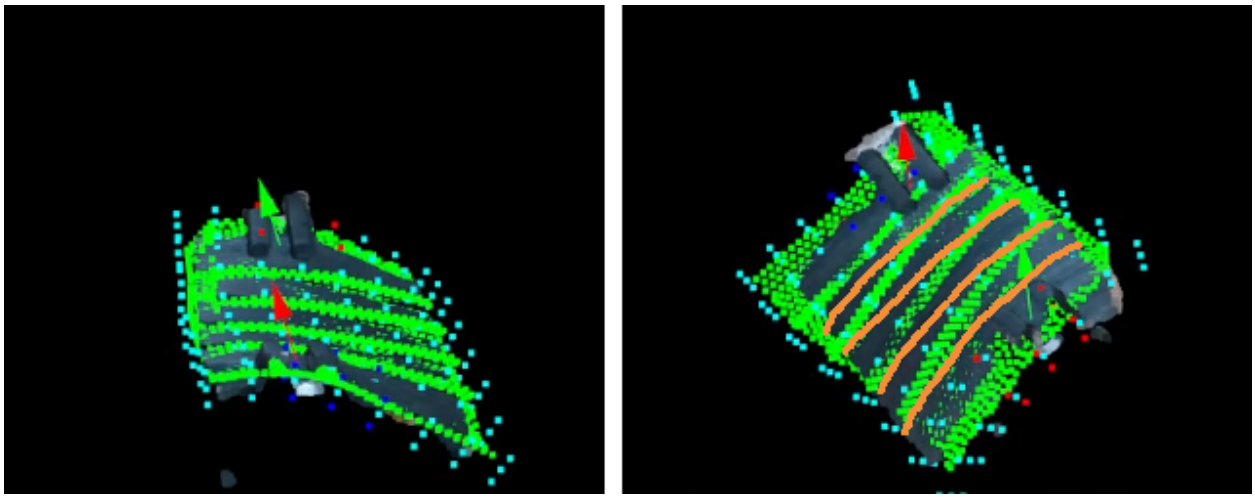

FIGURE 4.23: Visualisation du suivi de forme LARAP pour la tâche en cours

Gauche : Nuage de points déformé. A noter la précision de la déformation mesurée grâce au treillis, représenté par les points bleus. Droite : faiblesses de la méthode, manque de précision des points individuels. Les sillons, visibles pour l'humain dans le nuage de point, sont surlignées en orange.

Méthode propre à la tâche : détection de caractéristiques

Nous avons implémenté une méthode basée sur la détection de caractéristiques, dont l'interface est montrée dans la Fig. 4.24, l'implémentation est effectuée grâce à openCV [4], et les résultats montrés dans la Fig. 4.25.

La méthode est très performante et répétable dans la configuration initiale. Cependant, le fonctionnement interne de cette méthode nécessite l'usage de seuils. Nous n'avons pas réussi à filtrer assez efficacement l'image, ni réussi le paramétrage des seuils dynamiques pour effectuer toute la tâche avec cette méthode. La méthode reste parfaitement adaptée à l'acquisition de points cibles. Nous avons également essayé cette méthode sur les données provenant uniquement du champ "profondeur" de la caméra, sans succès : La résolution du champ profondeur n'était pas suffisante pour obtenir un résultat en deça des tolérances à atteindre, malgré nos efforts.

4. https://docs.opencv.org/3.4/d9/db0/tutorial_hough_lines.html

```
#ifndef FEATURE_DETECTION_H
#define FEATURE_DETECTION_H

#include "../common/genericLogic.hpp"
#include <opencv2/opencv.hpp>

// cette fonction a pour but de nettoyer la memoire des positions courantes des ribs et target.
//A appeler au premier call d'Init de la sequence
void vp_reset(void);
//Cette fonction detecte la mauvaise soudure de depart, les ribs, et calcule les points de saisie.
void vp_init(cv::Mat colorFrame, cv::Mat depthFrame);
void vp_idle(cv::Mat colorFrame, cv::Mat depthFrame);

std::vector<pose> vp_getGraspPoints(void); //retourne les points de saisie détectés plus haut
// Detecte la position de la partie mobile du thread, et celle de la soudure quand elles sont assez dissociées.
void vp_trackMovingEdge(cv::Mat colorFrame, cv::Mat depthFrame, bool isTaskPerform = false);
std::vector<pose> vp_getTargetPoints(void); // Retourne les points cibles
std::vector<pose> vp_getCurrentPoints(void); // Retourne les points de contrôle
cv::Mat vp_getAnnotedFrame(void);

#endif
```

FIGURE 4.24: Interface de la détection de caractéristiques

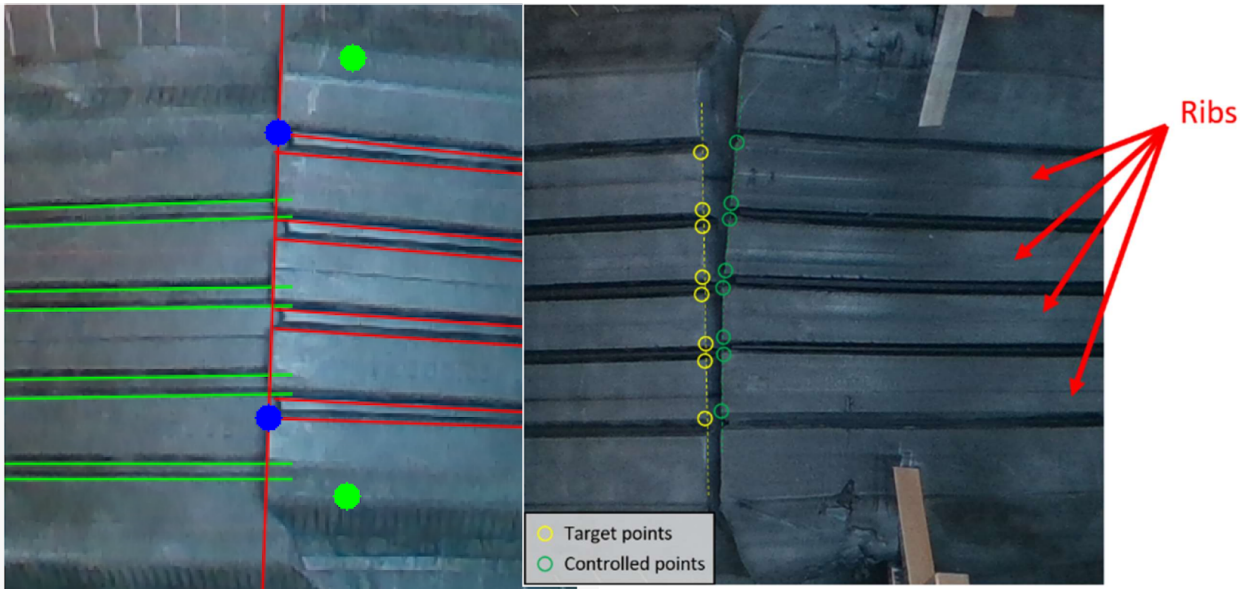

FIGURE 4.25: Résultat de la détection de caractéristiques

Gauche : image augmentée des lignes détectées et point de saisies. Droite : image augmentée des points cibles et des points de contrôle

Méthode par réseau de neurones : UNet

FIGURE 4.26: Procédé d'entraînement d'UNet

Gauche : Image issue du flux vidéo, rognée pour conserver uniquement la zone d'intérêt. Droite : masque dessiné à la main.

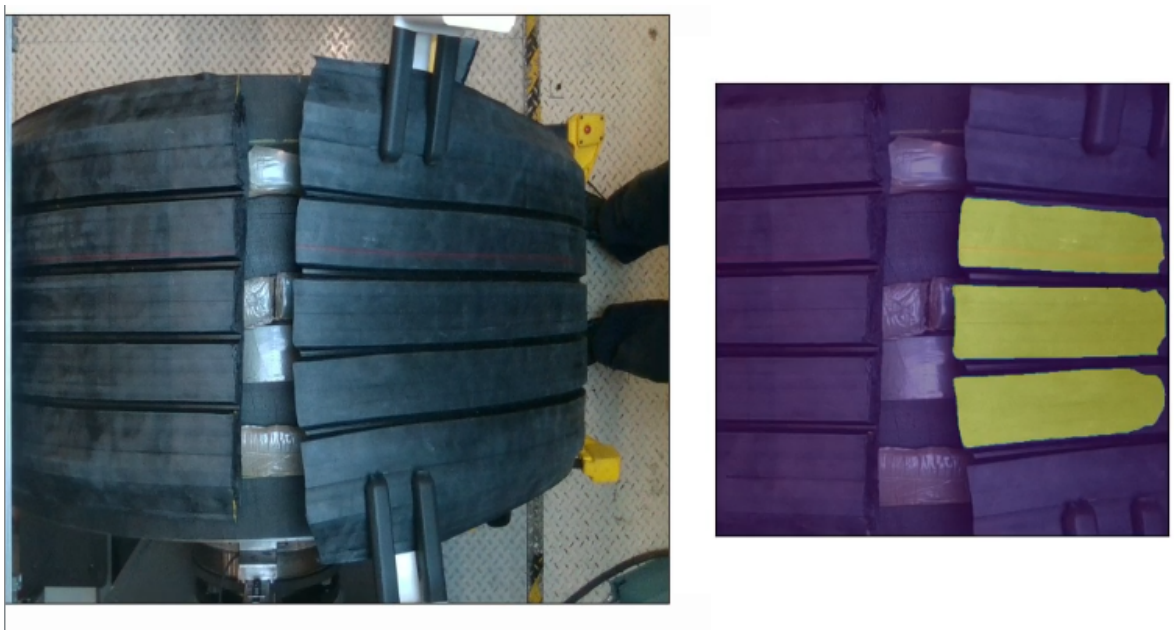

FIGURE 4.27: Résultat en direct de la perception basée sur UNet

Gauche : Flux vidéo. Droite : UNet appliqué à la zone d'intérêt.

Nous avons testé plusieurs méthodes basées sur des Réseaux de Neurone, détaillées dans l'annexe C. Cette section présente la plus efficace d'entre elles.

Cette méthode a été mise au point par Nicolas Roca Filella, Docteur-Ingénieur de Michelin. Il a utilisé le réseau de neurones UNet, entraîné sur une image RGB pour détecter les sillons de la bande

de roulement. Avec UNet, nous récupérons une image binaire avec les pixels de sculptures valant 1, les pixels de sillons ou de fond valant 0. Des fonctions de traitement d'image de base permettent d'extraire les points d'intérêts utilisés pour le contrôle. Ma contribution a été de participer à la labellisation des images, d'interfacer le code avec `Hades` via des socket ZMQ, et le tester de manière approfondie sur le cas d'usage.

Labellisation : Le réseau de neurones est appliqué sur une région d'intérêt. Nous avons pris des images dans des cas de fonctionnement, cadré sur la zone d'intérêts, grâce à un script Python développé par Nicolas Roca Filella. Ces images ont été ensuite labelisées à la main avec le logiciel Gimp, comme montré sur la Fig. 4.26. Les images ont été prises avec plusieurs bandes de roulement, de plusieurs profils, ainsi que plusieurs grippers, pour éviter d'overfitter le cas d'usage. Le réseau est ensuite entraîné.

Post-traitement : L'image est capturé dans le module Perception grâce au driver caméra. Elle est rognée, puis envoyée au programme Python tournant avec Pytorch dans un environnement Conda hors ROS. La communication se fait encore une fois avec ZMQ. Côté Python, l'image est passée dans le réseau UNet, puis filtrée. Ensuite, une fonction de détection de contours permet d'identifier les points d'intérêts. Cette séquence est présentée dans l'Alg. 11. Les contours filtrés sont renvoyés au nœud ROS `Perception`. Dans la callback de la Socket, les pixels obtenus sont déprojetés en position, formant un vecteur qui sera transmis aux autres nœuds.

Algorithme 11 Implémentation de la détection UNet

1: *crop* = *color_image*[100 :600, 400 :900]
2: *mask* = `UNet_predict`(*crop*)
3: *contour* = `findContour`(`morphoOpen`(*mask*))
4: **return** *contour*.`sort`()

L'intérêt du driver caméra permettant de fournir l'image sans la consommer à des clients différents prend tout son sens dans ce cas. En effet, l'image est à la fois augmentée et streamée à la perception, et coupée et envoyée au code Python y appliquant UNet.

```
1  #ifndef ARUCO_DETECTION_H
2  #define ARUCO_DETECTION_H
3
4  #include <opencv2/opencv.hpp>
5  #include "../common/genericLogic.hpp"
6
7  #include <opencv2/opencv.hpp>
8
9  void un_getPose(std::vector<int> param);
10 std::vector<pose> un_getCurrentPose(void);
11 std::vector<pose> un_getGraspPoint(void);
12 cv::Mat un_getframe(void);
13 void un_setFrame(cv::Mat);
14
15 #endif
```

FIGURE 4.28: Interface de la perception UNet

Intégration dans `Hades` : Tous les détails techniques ci-dessus sont abstrait à l'utilisateur. L'interface, présentée Fig. 4.28, permettant de calculer les positions du vecteur de pixels détecté avec la fonction ligne 9. Ligne 10, la fonction `un_getCurrentPose` permet de créer un vecteur

de pose exploitable par l'algorithme de contrôle. Ligne 11, la fonction `un_getGraspPoint` retourne les points de saisie de la gomme, calculés en fonction de la position des points détectés. Pour un retour visuel, la fonction `un_setFrame` permet de copier la frame envoyée au réseau en mémoire ; la fonction `un_getFrame` retourne la frame augmentée des points détectés.

Synthèse du test sur le cas d'usage réel : La méthode de perception par réseau de neurones a pour principale faiblesse des débordements : lorsque les deux couches sont proches, le réseau ne parviens pas à identifier de manière précise à quelle couche appartient le pixel. Malgré ce problème, potentiellement solvable avec plus d'entraînement ou des filtres, cette méthode de perception permet un asservissement de forme avec une précision dans nos marges de tolérance.

4.5.4 Modules de DeformationControl

Stratégies en boucle ouverte

Dans cette catégorie de stratégies, nous essayons de résoudre la tâche en reproduisant un mouvement effectué par un opérateur. Ces stratégies sont dites "boucle ouverte" car nous ne corrigeons pas la trajectoire avec les informations remontées par la perception.

Trajectoire manuelle

La manière la plus simple consiste à enregistrer quelques points cibles, et les rejouer. Les points cibles sont obtenus en plaçant les robots aux positions voulues, qui sont sauvegardées. Cette stratégie est très utilisée dans l'industrie, car extrêmement robuste et répétable – pour le moins dans le point de vue du robot. Cette stratégie est utilisée dans le banc de test final, pour les tâches Init et Stop. Nous avons enregistré des configurations articulaires initiales, intermédiaires et finales. Ces configurations ont été choisies pour éviter les problèmes de singularités et de collisions, non triviaux avec le placement des robots contraints par la mécanique existante.

Apprentissage par démonstration

Cette technique a été développée par Carlos Vélez García, ingénieur à Insescop.

Le principe est d'utiliser une LED et une centrale inertielle, montés sur une reproduction imprimée en 3D du préhenseur : avec une caméra RGBD, il est possible de suivre la position de la LED, tout en enregistrant les orientations provenant de la centrale inertielle - et reconstruire la trajectoire complète d'une manipulation humaine, pour la reproduire à l'identique avec un robot.

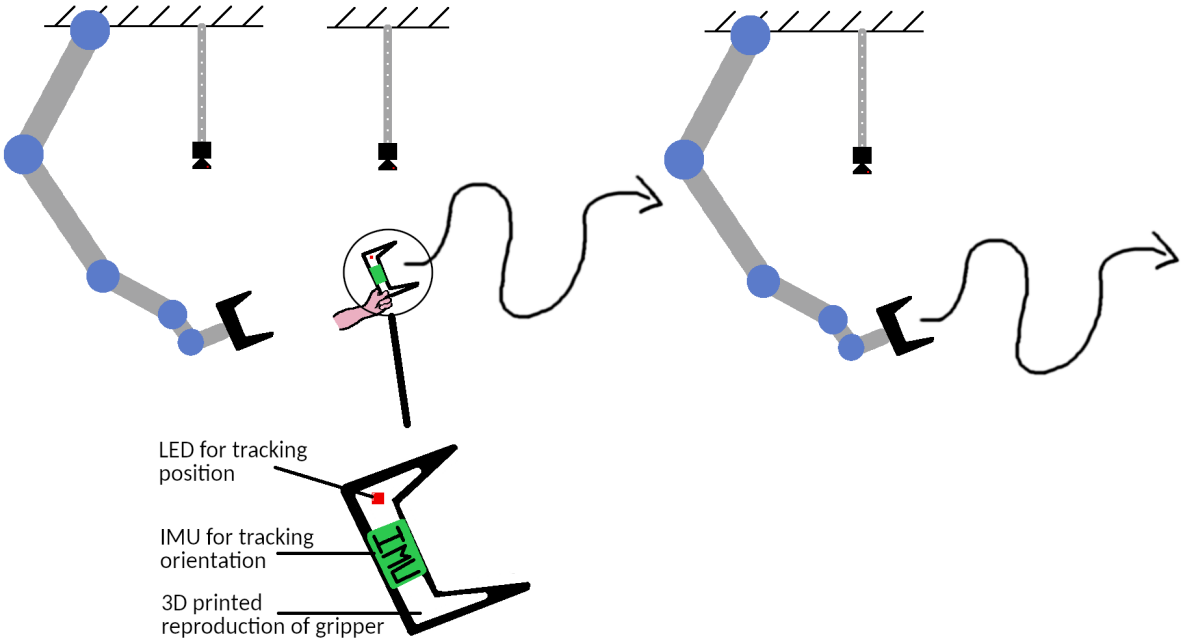

FIGURE 4.29: Procédé d'apprentissage de trajectoire par démonstration

Stratégie en boucle fermée : Contrôle proportionnel spécifique

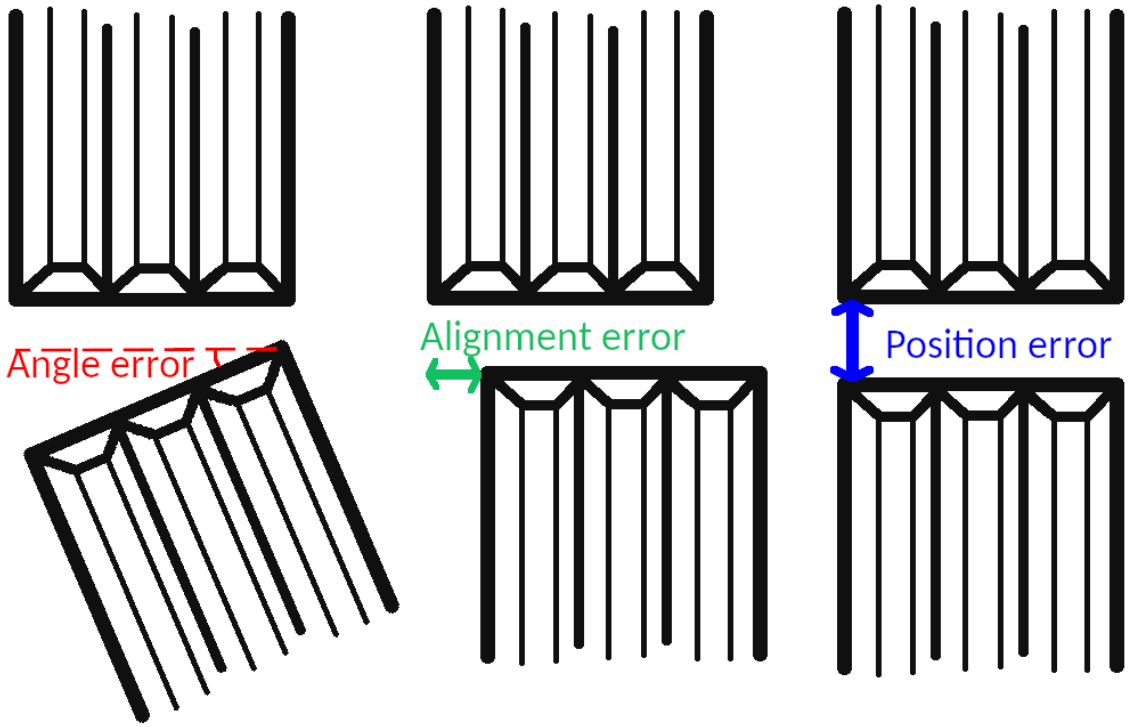

FIGURE 4.30: Erreurs indépendantes de contrôle du contrôleur spécifique

Nous avons tenté une stratégie simpliste, réduisant encore l'ordre du problème. Vu que nous souhaitons aligner deux bandes autour d'un cylindre, que les bandes sont presque bien enroulées en début de tâche, que la gomme est suffisamment rigide, nous avons réduit l'ordre du problème

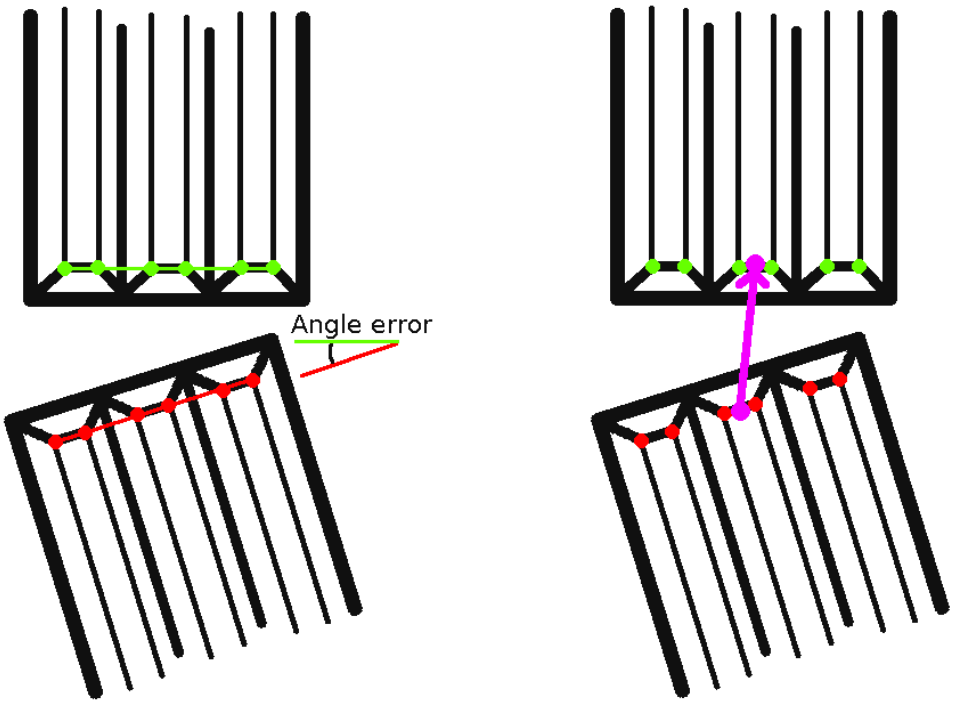

FIGURE 4.31: Obtention des erreurs du contrôleur spécifique

Gauche : calcul de l'erreur d'angle. Droite : en utilisant les points mesurés, il suffit de calculer l'isobarycentre des deux lignes de points pour obtenir un vecteur d'erreur. Le moyennage des points permet de lisser les erreurs de mesure.

```
#ifndef CONTROLLER_PROPORTIONAL_HPP
#define CONTROLLER_PROPORTIONAL_HPP

#include "../../common/mesh.hpp"
#include <Eigen/Dense>
#include "../../common/genericLogic.hpp"

void Controller_init(void);
void setControllerDesiredShape(std::vector<pose> targetShape);
void setControllerCurrentShape(std::vector<pose> currentShape);
void setRobotsCurrentPose(pose arm1Pose, pose arm2Pose);
std::vector<pose> controller(void);

#endif
```

FIGURE 4.32: Interface du contrôle spécifique

à 4 paramètres : l'erreur d'angle, l'erreur d'alignement, l'erreur de position et l'erreur de hauteur, illustrées dans la Fig. 4.30. Ces quatre erreurs nécessitent une perception capable de détecter non pas un maillage 2D, mais simplement une ligne de points, justifiant le développement des méthodes de perceptions propre à la tâche et UNet, détaillées dans la section 4.5.3. L'estimation des erreurs grâce à ces points est montrée Fig. 4.31

L'interface de ce contrôleur est présenté dans la Fig. 4.32. A noter que cette interface est similaire à celles des contrôleurs basés sur une Jacobienne ; les différences sont l'absence de modèle

FIGURE 4.33: Bande de roulement avec marqueurs ArUco pour valider la stratégie proportionnelle spécifique

externe et le type des données passées au contrôleur.

Avec une ligne de points cibles, et une ligne de points de contrôle, l'erreur d'angle peut être détectée avec des fonctions de trigonométrie. Les erreurs d'alignement, de hauteur et de position sont calculées avec des points virtuels, moyenne de tous les points de la ligne. L'utilisation de gains différents sur ces paramètres permet de prioriser les erreurs d'alignement et d'angle.

Ce contrôleur impose un mouvement sur l'axe de rotation parallèle aux sillons de la gomme, permettant d'obtenir un cintrage. Ce mouvement est effectué en boucle ouverte, mais la précision de la courbure de centrage n'est pas une priorité d'un point de vue procédé. Ce mouvement imposé permet de rapprocher fortement le mouvement effectué par le contrôleur de la trajectoire de déformation qu'un opérateur humain impose à la gomme, avec bien moins de complexité qu'une commande de type Optimal Shape Servoing.

Cette solution a été protoypée avec des marqueurs ArUco, montrés dans la Fig. 4.33. La méthode de perception avec marqueurs a été bien sûr implémentée dans le module de perception. Cette solution temporaire a permis de valider l'approche, et de lancer l'orientation des travaux de perception sur une méthode capable de remonter moins de points, mais de manière plus précise. Une démonstration de cette stratégie d'asservissement de forme, associée à UNet et l'IHM, est disponible sur `https://youtu.be/5mxQn5wVNyc`.

4.5.5 Conclusion

Tous les modules dont le transfert a été effectué sont des preuves de l'efficacité d'`Hades` et de sa valeur en tant qu'outil pour développer, prototyper et industrialiser des cas d'usages liés à la manipulation d'objets déformables.

4.6 Comparaison des modules développés

Dans cette section, nous étudions l'application et l'adéquation des modules détaillés dans les sections 4.4 et 4.5 à la tâche avec les métriques définie dans la section 1.1.3. En plus de l'analyse

de l'application des modules à un cas d'usage, cette section met en avant le support apporté par Hades, qui nous a permis de tester et comparer tout ces modules.

4.6.1 Comparaison des méthodes de perception

Quelle que soit la méthode utilisée, la précision dépendra de la résolution du capteur de vision : un pixel n'est pas une unité de distance absolue. Une erreur d'un pixel n'aura pas le même impact, comme illustré dans la Table 4.3 et la Fig. 4.34. La précision impacte également la vitesse d'exécution des algorithmes de fusion de données.

Résolution	Taille d'un pixel
240p	1.91mm
480p	0.96mm
720p	0.64mm

TABLE 4.3: Précision par pixel par résolution à hauteur de gomme dans le cas d'usage

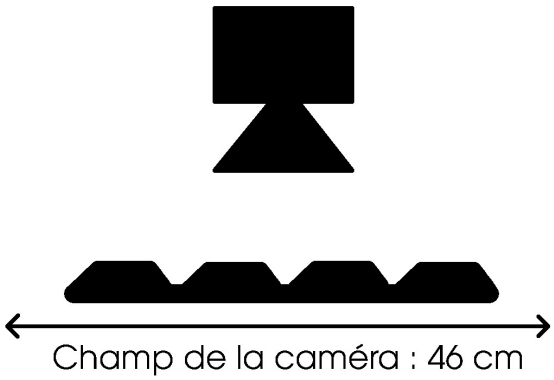

FIGURE 4.34: Visualisation du setup

TABLE 4.4: Performance des méthodes de perception

Solution	Moyenne/Ecart-type du temps de cycle	Précision mesurée
ROBUSfT (240p)	28,8ms / 5,8ms	2mm (Pixel)
ROBUSfT (480p)	62,7ms / 3,6ms	1mm (Pixel)
ROBUSfT (720p)	163,4ms / 1,4ms	0.5mm (Pixel)
LARAP (240p)	45,4ms / 0,5ms	5mm
LARAP (480p)	98,9ms / 0,8ms	8mm (décrochage)
LARAP (720p)	260,3ms / 1,1ms	12mm (décrochage)
UNet (720)	112,6ms / 12,1ms	1mm (débordement)

Les performances des différents algorithmes de perception sont résumées dans la Table 4.4. Il est préférable que le temps de cycle soit en dessous de 100ms – le délai devient apparent au delà.

— Dans le cas de ROBUSfT, le meilleur compromis se situe à une résolution de 480p - précision correspondant aux besoins du projet avec 1mm de bruit isotrope, temps de cycle suffisant pour permettre un bon contrôle. La méthode nécessite cependant de la texture, ce qui ne permet pas son application dans le cas d'usage.

— Dans le cas de LARAP, un temps de cycle plus élevé implique des décrochages quasi-systématiques à 480p et 720p, ce qui impact fortement la précision mesurée. Une vitesse de cycle permettant un suivi satisfaisant n'est possible qu'à 240p. Le patron utilisé n'a que peu d'influence sur la qualité du suivi tant qu'il est assez proche - par exemple, nous n'avons pas mesuré de détérioration ou d'amélioration significative en passant d'un patron d'une gomme avec 4 sillons à une avec 5 pour une gomme avec respectivement 5 et 4 sillons

— Dans le cas d'UNet, le délai est exacerbé par la couche de communication nécessaire supplémentaire à la solution développée. Cependant, les résultats obtenus restent acceptable. La précision mesurée est satisfaisante : en latéral, là où les tolérances sont les plus basses, l'erreur est en dessous du millimètre 9 fois sur 10. L'erreur est proche du millimètre longitudinalement. Cela est dû au phénomène de débordement : lorsque la soudure est presque réalisée, le réseau discerne difficilement si un pixel de gomme appartient à la bande mobile ou la bande statique. Cette direction est celle où nous avons le plus de marge d'erreur, ce qui rend la solution particulièrement adaptée à la tâche.

TABLE 4.5: Récapitulatif des méthodes de perception appliquées à la tâche

Méthode	**Forces**	**Faiblesses**
ROBUSfT	Rapide Précis	Nécessite un objet texturé Nécessite un patron de la texture
LARAP	Exploite les données du champ profondeur d'une caméra RGB-D Ne nécessite pas de texture	Nécessite un objet déjà saisi Décroche facilement
UNet	Précis Ne nécessite pas de texture	Nécessite de la labellisation

4.6.2 Comparaison des stratégies de contrôle en boucle ouverte

Les différentes stratégies en boucle ouverte testées sont comparées dans la table 4.6. Ces stratégies fonctionnent bien mieux pour les tâches d'extraction et d'insertion que de soudure précise : la plus haute déformabilité des deux pièces permet de conclure la tâche en forçant avec un mouvement rotatif à une précision de l'ordre du demi-centimètre, qui est au delà des tolérances pour le cas d'usage du pneu. De plus, les pièces sont généralement plus petites et à des positions extrêmement répétables, diminuant le problème de saisie et améliorant la performance de ces stratégies.

TABLE 4.6: Récapitulatif des stratégies boucle ouverte appliquées à la tâche

Stratégie	**Forces**	**Faiblesses**
Calibration manuelle	Facilité de mise en oeuvre	Insuffisant pour les tâches complexes
DQ-DMP	Trajectoire adaptée aux robots Trajectoire adaptable à différentes configurations initiales et finale	Trajectoire filtrée ne conservant pas l'écart entre les préhenseurs
Dexterous learning	Apprentissage rapide d'une calibration manuelle	Même résultat qu'une très bonne calibration manuelle

4.6.3 Comparaison des stratégies de contrôle en boucle fermée

Le premier constat lié au stratégies de contrôle en boucle fermée : la précision finale de l'assemblage dépend bien plus de la perception que du contrôle, ce qui est attendu. Les différentes stratégies de contrôle en boucle fermée sont présentées dans la table 4.7.

TABLE 4.7: Récapitulatif des stratégies boucle fermée appliquées à la tâche

Stratégie	**Forces**	**Faiblesses**
Jacobien-proportionnel	Simplicité d'implémentation	Nécessite un modèle
Optimal Shape Servoing	Prise en compte de la trajectoire de déformation Apprentissage possible	Nécessite un modèle Difficile à comprendre
Proportionnel spécifique	Simplicité d'implémentation Adéquation à la tâche Nécessite peu de points	Manque de généricité

4.7 Discussion

Dans ce chapitre, nous avons présenté Hades, une architecture logicielle permettant de prototyper, tester, transférer et itérer sur des travaux de recherche et de développement portant sur la thématique de manipulation d'objets déformables. Le cas concret du projet SoftManBot a permis de mettre en avant la capacité d'Hades à s'adapter à des développements variés, ainsi que sa capacité à transmettre des résultats de recherche à l'industrie. Enfin, nous avons utilisé Hades comme support de comparaison des recherches développées dans le cadre du projet.

Hades remplit donc les besoins identifiés.

— Premièrement, nous pouvons aisément recompiler et ré-exécuter des manipulations sur des plateformes différentes.
— Deuxièmement, il est facile de changer une fonction identifiée sans avoir à impacter toute la base de code grâce à la modularité mise en oeuvre.
— Troisièmement, la flexibilité d'Hades permet des développements itératifs de type méthode Agile.
— Quatrièmement, nous n'avons pas de problème de licence.
— Cinquièmement, nous avons pu tester plusieurs algorithmes de perception et de commande, et les comparer objectivement grâce à la modularité de la base de code.

Hades et l'exposition des travaux sur le banc de test permet de donner une vision globale à quiconque souhaiterais implémenter une solution de manipulation d'objets déformables, permettant aux chercheurs de se comparer facilement à l'état de l'art et aux industriels de gagner en vitesse de prototypage.

Les limitations d'Hades sont liées à la base d'utilisateurs : plus il y aura d'utilisateurs présents, plus il y aura de modules développés et le plus l'architecture sera attractive. Nous nous sommes concentrés sur une application pour ROS-1, qui est désormais obsolète pour les problématique de recherche. Une migration pour ROS-2 est l'un des points prioritaires : cela permettrait d'assurer une continuité des développements et respecter les contraintes de temps réel, même si aucun problème lié à ce point n'a été identifié dans nos tests.

Les orientations futures des développement portent sur l'intégration de fonctionnalités de généralisation, pour permettre d'interfacer plus facilement l'apprentissage par démonstration soit d'IA, soit d'identification de paramètres. De plus, il est actuellement difficile de détecter les erreurs - et donc, d'autant plus de faire de la reprise sur erreur ou de notifier un opérateur lorsqu'émergent des problèmes. Enfin, intégrer Hades à des frameworks existants comme Robmosys ou PlanSys2 permettrait de toucher plus d'utilisateurs et d'intégrer une cellule robotique de manipulation d'objets déformables à un écosystème autosuffisant ROS à portée industrielle.

Les travaux de ce chapitre montrent pleinement la complexité de la manipulation robotique d'objets déformables, en particulier pour les problématiques d'assemblage. Hades a servi de support aux travaux de recherches présentés tout au long de la section, leur fournissant une confrontation avec un cas réel aux indicateurs clés connus et mesurables, permettant de répondre à la question suivante : quelle est la combinaison d'approche de modèle, la méthode de perception et la stratégie d'asservissement de forme la plus adaptée pour la tâche à accomplir ? Cette question possède bien entendu autant de réponses que de cas d'usage, mais il est désormais possible d'y répondre bien plus rapidement avec les modules développés pour Hades.

Conclusion

Synthèse des travaux

Dans ces travaux de thèse, nous avons présenté un cas concret de manipulation d'objet déformable, la soudure de bande de roulement. Non seulement nous avons résolu ce cas, mais nous avons également pu comparer divers moyen d'accéder à ce but, grâce à notre proposition d'`Hades`, architecture logicielle modulaire qui nous a permis de combiner différentes approches de modèle, méthodes de perception et stratégies d'asservissement de forme, de les appliquer au cas d'usage, et de déterminer objectivement la combinaison la plus adaptée au problème à résoudre. `Hades` formalise une séparation fonctionnelle de la mise en œuvre des différentes techniques décrites dans cet état de l'art en les organisant sous forme de modules. Cette contribution a permis de répondre à la question de la meilleure combinaison d'approches de modélisation, de méthodes de perception, et de stratégies de contrôle pour résoudre une tâche proposée

L'une des stratégies, la contribution Optimal Shape Servoing (`OSS`), a été explicitement développée pour résoudre la soudure de la bande de roulement.. `OSS` permet de générer implicitement une trajectoire de déformation sans avoir besoin d'imposer des formes intermédiaires. La stratégie provient de l'application de la stratégie de commande optimale LQG dont la formulation a été adaptée au cas de la manipulation d'objet déformable. Cette contribution propose une application innovante de la théorie de la commande optimale, en donnant un sens physique aux matrices de poids pour influer sur la trajectoire de déformation de l'objet manipulé. De plus, nous proposons une formalisation du problème permettant d'utiliser cette stratégie pour découpler l'erreur de forme et l'erreur de position.

`OSS` est également une prolongation de la stratégie proportionnelle ARAP-SS pour As-Rigid-As-Possible Shape-Servoing : au delà d'une itération sur le contrôle proportionnel utilisant ce modèle, la contribution proposée s'appuie sur la faisabilité de cette stratégie. Cette contribution a prouvé que la plus simple des stratégies de contrôle linéaire fonctionne pour le cas d'usage privilégié. Cette contribution répond à la question de l'adéquation de l'utilisation de stratégies de contrôle avancées pour ce cas d'usage - en particulier pour répondre aux contraintes propres à l'industrie comme le respect des temps de cycle et les normes de qualité.

Enfin, l'utilisation d'algorithmes génétiques permet d'effectuer un apprentissage par démonstration des paramètres des matrices de poids. Cette contribution pousse l'adaptabilité de la stratégie à son paroxysme, en bénéficiant à la fois d'une commande issue d'une stratégie connue et fiable et de l'adaptabilité à un problème donné permis par les techniques d'IA. La nouveauté de la stratégie provient de l'application de la commande optimale au problème de manipulation d'objet déformable, donnant un sens physique aux matrices de poids, permettant d'asservir non seulement la forme de l'objet, mais également la trajectoire de déformation de ce dernier. Cette stratégie permet également de prioriser de manière quantifiée l'erreur de forme et l'erreur de position sur certaines manipulations. Enfin, il est possible d'apprendre les paramètres de cette commande par démonstration.

Implications et limitations

Cette section présente des questions ouvertes, soulevées par nos recherches.

Degrés de liberté du système

La manipulation d'objets déformables est confrontée à des problèmes de commande de systèmes sous-actionnés. Mais à quel point ? La quantification exacte de ce sous-actionnement est loin d'être triviale.

Sur-estimation dûe aux dimensions du système : Une analyse théorique du système défini pour `OSS` dans la section 3.2 donne un nombre de DDL de $3n$, avec n le nombre de nœuds ; avec 3 DDL par nœud dans l'espace 3D. Le nombre de DDL des actionneurs sont de $6r$, avec r le nombre de robots. Le nombre 6 vient des 3 translations et des 3 rotations possibles. Une considération très pratique permet de réfuter cette approche : en effet, elle implique que le système peut avoir un de ses nœud translaté sur un axe à une distance arbitraire sans aucun impact sur les autres nœuds, ce qui est illustré Fig. 4.35.

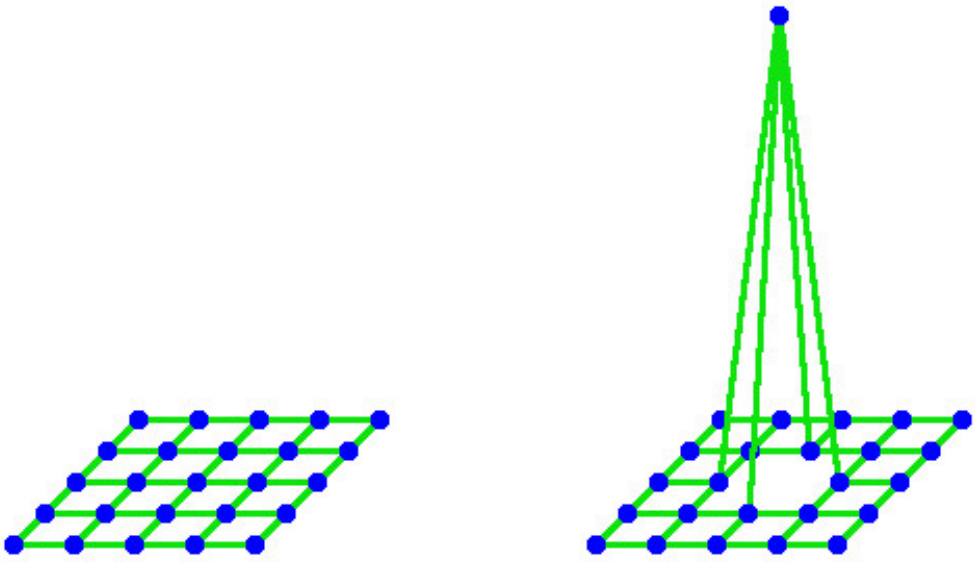

FIGURE 4.35: Illustration du biais de sur-estimation dûe aux dimensions du système

Gauche : Maillage au repos. Droite : sous l'hypothèse des 3n DDL, l'état formé par une translation en z d'un mètre du nœud du milieu est sensé être atteignable. Ce comportement n'est bien sur pas physiquement possible.

Même si ce DDL existe potentiellement, aucun positionnement des actionneurs ne permet d'atteindre cette configuration en statique - ce qui est d'autant plus vrai si les nœuds saisis sont au bord de l'objet et pas en son centre. Les DDL actifs sont uniquement les modes statiques atteignable par l'objet avec une configuration de saisie donnée.

Surestimation dû aux effets des préhenseurs Nous avons remarqué empiriquement que l'action de préhension modifiait les propriétés physiques des objets manipulés, en augmentant la rigidité locale et en imposant des contraintes dures de distance entre les nœuds. Ces deux phénomènes contribuent à la diminution du nombre de DDL réels du système.

La question de la quantification exacte de la sous-actuation du système reste ouverte, que l'on peut formuler de la manière suivante : quel est le degré de sous-actionnement au sens des DDL actifs dans les conditions quasi-statiques ?

Découplage des erreurs de forme et de position

Une nouvelle façon d'envisager les problèmes de manipulation d'objet déformable est de réguler une erreur de forme et une erreur de position, au lieu d'une erreur de forme globale. Ce découplage permettrai de synthétiser des contrôleurs avec des mouvements plus régulier et des déformations plus adaptées à l'objet, minimisant le risque de dommage. Les contrôleurs découplant l'erreur de forme et l'erreur de position sont plus intuitifs et prévisibles, permettant un meilleur réglage de leurs paramètres. Nous avons appliqué ce paradigme à la stratégie d'asservissement de forme `OSS` dans la section 3.4.3. Les résultats prometteurs impliquent de nombreuses applications pratiques aux méthodes existantes.

Couplage des approches basées robot et des approches basées objets

Les approches basées robots effectuent un mouvement prédéfini, avec un opérateur expert paramétrant le mouvement du robot. Les approches basées sur l'objet offrent la possibilité d'une commande et d'une trajectoire intelligente tenant compte de l'état de l'objet. Les deux approches présentent des avantages et des inconvénients.

Les avantages des approches basées sur les robots sont des mouvements sont souvent plus lisses ou présentant des particularités difficiles à modéliser. Les inconvénients sont que l'erreur de forme finale est bien supérieure à ce qui est souvent acceptable.

Les avantages des approches basées sur l'objet sont la précision supérieure de l'asservissement de la forme, à la fois en termes de déformation et de position. Les inconvénients sont la charge de calcul, la nécessité d'un modèle et l'impossibilité de transcrire des mouvements directs des effecteurs.

Les approches peuvent être fusionnées – simplement en utilisant une trajectoire générée et un contrôleur de forme classique. L'ordre final envoyé au robot peut être une combinaison de la trajectoire apprise et de la correction de forme ou de position. Ce couplage permettrait d'obtenir le meilleur des deux mondes, d'une manière similaire à ce qui est proposé pour le découplage des erreurs de forme et de position, en particulier pour ce qui est de la gestion des rotations 3D des préhenseurs.

Perspectives

Nos travaux de recherche ouvrent plusieurs perspectives. Tout d'abord, en termes de modèle, nous souhaitons travailler sur une formulation de type éléments finis appliqué à un objet linéique. Les résultats préliminaires montrent une modélisation rapide prenant en compte la pré-charge de la structure en utilisant des éléments à 2 nœuds pour les effets de traction/compression et à 3 nœuds pour les effets de flexion. Ce modèle, basé sur des équations physiques et non des éléments géométriques, permettrait de tester la commande `OSS` sur un modèle différent, et de quantifier l'impact de la précision du modèle sur la précision de la commande.

Ensuite, nous souhaitons explorer la piste de l'identification des matrices de commande robuste par démonstration. Une première étape sera la validation sur un système réel, et la comparaison avec d'autres méthodes d'identification comme la descente de gradient. Nous envisageons également de développer un contrôleur contenant une pondération entre Dual-Quaternion based Dynamic Movement Primitives et ARAP-SS - le pourcentage de répartition envisagée correspond à la fonction de répartition de la loi normale, pour commencer la tâche en privilégiant les mouvements à effectuer et la finir en privilégiant l'asservissement de la forme de l'objet.

Enfin, au sujet d'`Hades`, du travail de maintenance et d'extension demeure. Basculer l'architecture sous ROS 2 sera nécessaire pour intégrer les futurs développements académiques. L'architecture est potentiellement intégrable à d'autres architectures plus généralistes, comme PlanSys2 : il serait souhaitable de l'intégrer à cette architecture plus générique et de plus haut niveau. Pour finir, une architecture est un outil, il faudrait donc s'en servir pour les prochains développements au sein du laboratoire et des prototypes industriels afin de faire évoluer l'écosystème développé.

Annexe A

Préoccupations éthiques

Non seulement la sensibilisation éthique est obligatoire pour les doctorants, mais cela est également fortement lié à mes valeurs. L'éthique entre en intersection avec l'épistémologie, qui à son tour impacte la vision du monde des chercheurs. Ainsi, expliciter ma vision du monde aide à contextualiser mes recherches.

Préoccupations éthiques générales

Mon postulat épistémologique est l'empirisme, fortement influencé par l'inférence bayésienne et le mouvement sceptique.

Empirisme L'empirisme s'oppose au rationalisme, qui tente de voir le monde à travers la logique et la pensée, en construisant un modèle complet et en déduisant la vérité uniquement à partir des pensées. Je crois que la vérité est quelque chose de défini comme testable. J'apprécie les idées, la raison et les instincts pour générer de nouvelles idées, mais je préfère qu'elles soient testables dès que possible.

Inférence bayésienne L'inférence bayésienne est un postulat épistémologique issu de la statistique. Le monde n'est pas considéré avec des certitudes, mais plutôt avec des pourcentages de confiance. Plus j'ai de preuves, plus ma confiance grandit. Cependant, de nouvelles preuves, hypothèses ou - plus impactantes - des preuves allant à l'encontre de ma vision du monde doivent changer ma vision du monde si je les accepte comme valides. Cela est difficile, surtout avec des croyances que l'on apprécie.

Mouvement sceptique La principale leçon que j'ai tirée du mouvement sceptique et intégrée dans mon processus de pensée critique est : les affirmations extraordinaires nécessitent des preuves extraordinaires. Cela est particulièrement valable dans le monde de la recherche, où chaque article peut être considéré comme une affirmation extraordinaire d'avoir accompli quelque chose de nouveau.

Autres influences philosophiques En dehors des empiristes, deux philosophes ont eu un impact sur ma vision du monde. Tout d'abord, Spinoza et son déterminisme : Tout, et tout le monde, agit en raison de causes qui les déterminent, ce qui signifie que le libre arbitre est une simple illusion. Cependant, en tant qu'empiriste, je ne crois pas que cela soit une vérité, car cela n'est pas prouvable. Pour rationaliser cette croyance déprimante, j'utilise le pari de Pascal. En remplaçant

Dieu par le libre arbitre dans cet expérience de pensée, cela conduit à mon choix de croire au libre arbitre (ou d'être prédéterminé par mes causes en croyant au libre arbitre).

Ces influences définissent ma vision du monde et ont donc un impact sur mon travail à la fois en tant que chercheur et ingénieur logiciel, notamment dans ma méthodologie de développement logiciel. La force de l'empirisme réside dans l'apprentissage non seulement de ses propres erreurs, mais aussi des erreurs des autres, ce qui est cohérent avec la démarche de tout thésard devant se plonger dans l'exercice fastidieux de revue de l'état de l'art.

Serment d'Hippocrate de David King pour les scientifiques :

David King, conseiller scientifique en chef du gouvernement britannique, a proposé la généralisation suivante du serment d'Hippocrate pour toutes les professions scientifiques. Le code est organisé en trois sections et regroupe sept principes :

Rigueur

— Agir avec compétence et soin dans tout travail scientifique. Maintenir des compétences à jour et contribuer à leur développement chez les autres.
— Prendre des mesures pour prévenir les pratiques corrompues et les fautes professionnelles. Déclarer les conflits d'intérêts.
— Être attentif aux manières dont la recherche découle et affecte le travail d'autres personnes, et respecter les droits et les réputations des autres.

Respect

— Veiller à ce que votre travail soit légal et justifié.
— Minimiser et justifier tout effet négatif que votre travail pourrait avoir sur les personnes, les animaux et l'environnement naturel.

Responsabilité

— Chercher à discuter des problèmes que la science soulève pour la société. Écouter les aspirations et les préoccupations des autres.
— Ne pas délibérément tromper, ni permettre à d'autres d'être trompés, en ce qui concerne les questions scientifiques. Présenter et examiner de manière honnête et précise les preuves scientifiques, les théories ou les interprétations.

Serment de l'école doctorale :

L'école doctorale demande aux doctorants de prêter serment sur le plan éthique à la fin de leur doctorat [1]. Ce serment est entré en vigueur cette année.

« En présence de mes pairs. Parvenu à l'issue de mon doctorat en Robotique, et ayant ainsi pratiqué, dans ma quête du savoir, l'exercice d'une recherche scientifique exigeante, en cultivant la rigueur intellectuelle, la réflexivité éthique et dans le respect des principes de l'intégrité scientifique, je m'engage, pour ce qui dépendra de moi, dans la suite de ma carrière professionnelle quel qu'en soit le secteur ou le domaine d'activité, à maintenir une conduite intègre dans mon rapport au savoir, mes méthodes et mes résultats ».

1. Arrêté du 26 août 2022 modifiant l'arrêté du 25 mai 2016 fixant le cadre national de la formation et les modalités conduisant à la délivrance du diplôme national de doctorat, `https://www.legifrance.gouv.fr/jorf/id/JORFTEXT000046228965`

English version : « In the presence of my peers. Having completed my doctorate in Robotics, and having thus practiced, in my quest for knowledge, the exercise of a demanding scientific research, by cultivating intellectual rigor, ethical reflexivity and in the respect of the principles of scientific integrity, I commit myself, for what will depend on me, in the continuation of my professional career whatever the sector or the field of activity, to maintain an honest conduct in my relationship to knowledge, my methods and my results »

Comme ce serment formel est plus ouvert à l'interprétation, j'estime que l'ajout du serment de David King est nécessaire, du moins pour définir ce qu'est une "conduite honnête".

Préoccupations éthiques de la robotisation de la main-d'œuvre

L'objectif à long terme de ce travail est l'automatisation du travail manuel. Analysons-le à la lumière du serment de David King, en particulier les sections Respect et Responsabilité, car la section Rigueur est générale.

Respect

— Le travail est légal, financé par les programmes de recherches de l'Union Européenne elle-même.
— Il y a peu ou pas d'effet sur les animaux et l'environnement naturel. Cependant, des questions se posent quant à l'éloignement au travail des personnes.
— Le travail automatisé est fastidieux et comporte des risques tels que des troubles musculosquelettiques dus à la répétition des mouvements et l'exposition à des produits chimiques nocifs.

Responsabilité

— Même si le travail est difficile et dangereux, est-il éthique de le retirer aux personnes ? Les propositions et les plans d'affaires des partenaires industriels reposent sur la transformation d'un travail non qualifié et difficile en opérateurs valorisés supervisant des machines. Cette dernière option est meilleure pour l'image de soi des travailleurs.
— Dans notre monde capitaliste mondialisé, il y a une course à la productivité. Ne pas utiliser la robotisation ne ferait que retarder l'Europe et risquer des délocalisations. Une usine délocalisée supprime des emplois de chauffeurs routiers, de ressources humaines, de responsables logistiques et de directeurs d'usine. La robotisation peut éviter cela et peut-être même permettre la relocalisation des usines.
— Cependant, une dernière préoccupation concerne la redistribution des profits. Une entreprise qui continue de croître tout en remplaçant une partie de sa main-d'œuvre par des robots creusera l'écart entre les riches et les pauvres. En tant que société, nous devons nous préoccuper des modèles de redistribution pour éviter les effets indésirables.

Annexe B

Stratégie de saisie de la gomme

Dans le cas d'usage, la bande de gomme manipulée est placée au sommet de l'assemblage du pneu. La bande de roulement est bien plus épaisse que les autres nappes formant le produit, et l'assemblage fait qu'il y a suffisamment de place pour que l'opérateur puisse glisser ses doigts entre les bandes. Nous avons conçu une trajectoire d'effecteur capable de saisir la gomme, tout en adaptant la position du point de saisie et l'orientation du préhenseur avec des retours visuels. La séquence de saisie peut être décomposée en étapes, représentées dans Fig. B.1.

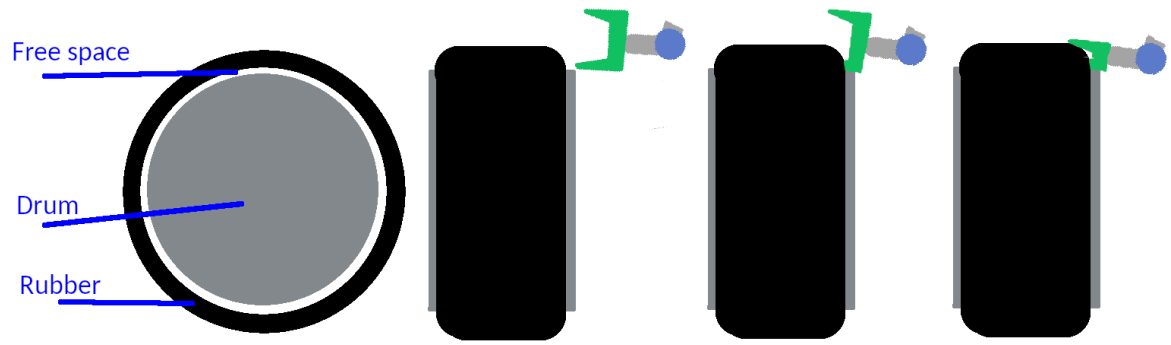

FIGURE B.1: Décomposition de la trajectoire de saisie

Etape 1 : approche. Etape 2 : Insertion. Etape 3 : Fermeture des doigts.

La première étape est l'approche. Une partie est faite avec un contrôleur articulaire, pour maîtriser la configuration du robot en début de tâche et éviter les singularités, dans la tâche Init. A quelques centimètres du pneu, le contrôleur utilisé devient un contrôleur cartésien. En utilisant les données de la perception, nous adaptons la hauteur et l'orientation du préhenseur.

La seconde étape est l'insertion. Les derniers centimètres suivent une trajectoire droite allant directement aux points de saisie détectés par la perception.

La troisième étape est la fermeture des doigts. Une fois la gomme saisie, les robots la soulève avant l'étape de soudure.

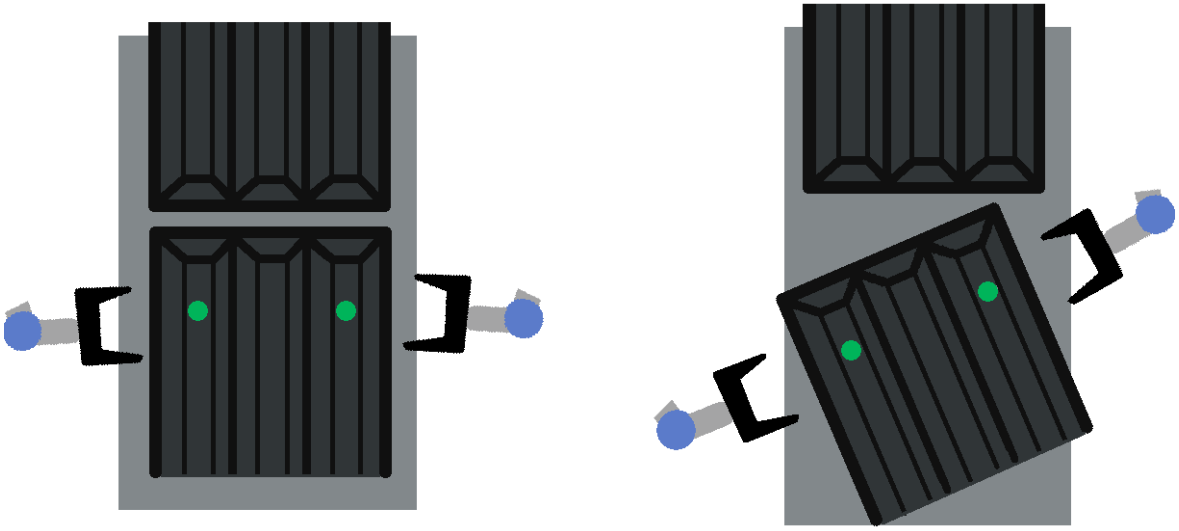

FIGURE B.2: Adaptation de la trajectoire de saisie grâce aux inputs de perception

Vert : points de saisie détectés. La trajectoire est adaptée dès l'étape d'approche.

De l'importance des formes de doigt

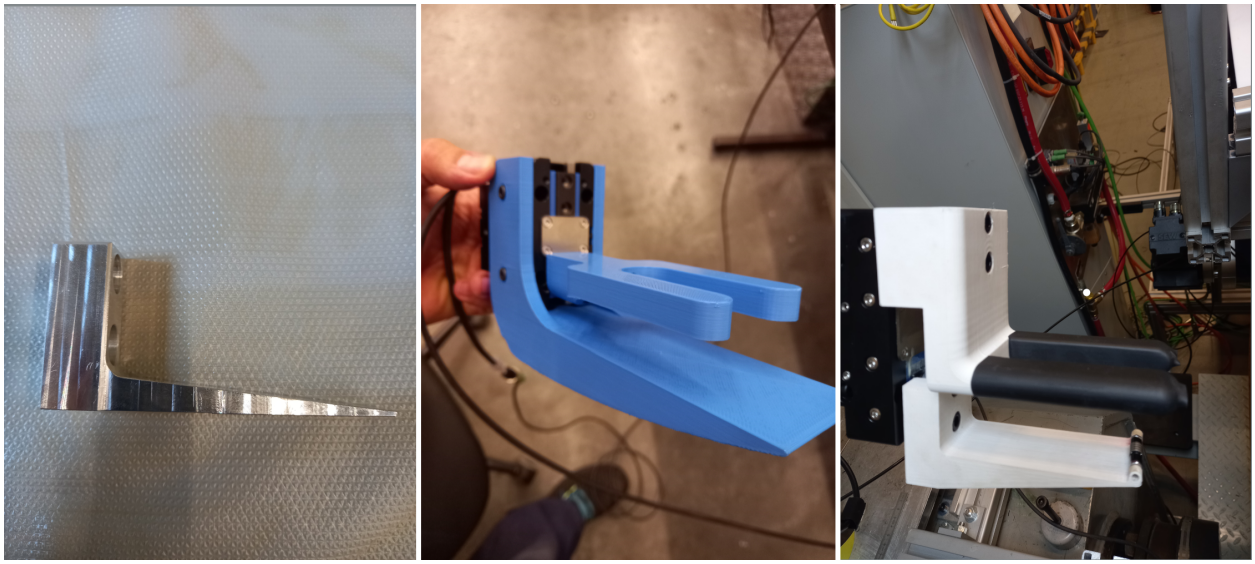

FIGURE B.3: Design de quelques itérations des doigts

Gauches : doigts poignard. Milieu : doigts imprimés en 3D avec de longues paumes et des doigts arrondis. Droite : doigts imprimés en 3D avec rouleaux et doigts arrondis

Une approche logicielle intelligente n'est pas une solution miracle capable de résoudre n'importe quel problème. Lors de cette tâche de saisie, nous avons du surmonter plusieurs challenges matériels.

— Tout d'abord, les préhenseurs eux-même. Lors de la phase de design, nous avons rejeté les préhenseurs basé sur des ventouses, comme notre objet est trop lourd. Nous avons rejeté les préhenseurs à aiguille, pour éviter d'endommager l'objet. Nous avons rejeté les préhenseurs magnétiques, car la gomme de la bande de roulement ne contient pas de métal.

FIGURE B.4: Impact de la forme des doigts sur l'intégrité de la gomme

Gauche : bande de roulement manipulée par les poignards. Droite : bande de roulement saisie par les doigts imprimés en 3D. Les deux bandes de roulement ont servies à des tests durant 1 mois.

Reste les pinces. Plusieurs modèles ont été testés. Chronologiquement, il s'agit de pinces électriques du partenaire Zimmer, de pinces 2-doigts et 3-doigts de Robotiq compatible avec ROS, et de pinces pneumatiques du partenaire Zimmer. Les modèles électrique du catalogue de Zimmer sont conçues pour toujours saisir au même écart. Le cas échéant, elles se mettent en défaut, ce qui est fréquemment arrivé avec la gomme manipulée. De plus, la pince manquait de couple avec cette mécanique de distance constante et la gomme glissait systématiquement. Les modèles compatible ROS était bien pour prototyper, mais trop fragile pour l'environnement industriel. Au final, les pinces pneumatiques tout-ou-rien se sont révélées être le meilleur choix.

— Ensuite, les doigts eux même. Le design de base des doigts Robotiq, des palets rectangulaires de 3cm par 5cm était intéressant, bien qu'un peu petits. Nous avons essayé des doigts métalliques plus long, car nous souhaitons saisir les épaules de la bande de roulement et non pas les ailettes. Cependant, les arêtes saillantes et la zone de contact réduite sur les saisies non parfaites ont valus à ce design le surnom de "poignards". Ainsi, nous avons imprimé en 3D plusieurs itérations des doigts. Nos tentatives nous ont enseignées que la partie inférieure doit être plate pour permettre une meilleure insertion, et la partie supérieure arrondie pour éviter d'endommager la bande de roulement. Fig B.3 montre plusieurs itérations de formes de doigt. Nous avons incorporé dans la partie plate un cylindre monté avec des roulements à bille. Ce cylindre roule sous la gomme, permettant de la préserver. Pour améliorer la qualité de la saisie, nous avons couvert les doigts supérieurs avec de la gaine thermodynamique. En effet, les doigts sont en Polyactide (PLA), et la friction PLA/gomme est relativement basse alors que la friction gomme/gomme est élevée. Ceci permet d'améliorer significativement la qualité de la saisie.

L'ensemble de ces décisions a drastiquement amélioré la qualité de notre saisie, capable à la fois d'effectuer la tâche et de préserver la bande de roulement, comme le montre la Fig. B.4.

Annexe C

Approches prometteuses et infructueuses

Perception basée sur un réseau de neurones

Nous avons envisagé l'application d'un Deep Neural Network sur le flux RGB-D de la caméra pour résoudre le problème de perception dès le début du projet. Cette sous-section présente les approches essayées.

Vue d'ensemble des approches tentées

Nous avons tenté de mettre en place plusieurs méthodes, avant de réussir avec UNet comme présenté dans la section 4.5.3.

FIGURE C.1: Vue de la tentative de labellisation des points d'intérêts

L'idée était de suivre les points formant le bord des nervures en utilisant à la fois le cadre RGB et le cadre de profondeur. Nous avons configuré la caméra RGB-D avec une vue en diagonale plutôt que la vue de dessus généralement utilisée dans le projet pour cette tentative.
Ces images sont une reconstitution et ne proviennent pas de l'ensemble de données d'entraînement réel, qui n'était plus disponible au moment de la rédaction.

— Une première méthode a été mise en place, basée sur de l'apprentissage supervisé. Ces travaux ont été effectués par Naveen Skumaar, stagiaire à l'Institut Pascal. Ma contribution était de fournir des données brut à labelliser, ainsi que le support aval d'intégration si la méthode aboutissait. L'idée était d'utiliser le réseau de neurones pour suivre le bord déformé de la bande de roulement. La labellisation souhaitée est montrée Fig. C.1. Malheureusement, nous n'avons pas eu assez de temps dans la durée du stage pour labelliser suffisamment d'image dans des contextes pertinents pour que le réseau soit suffisamment précis.
— Une deuxième méthode utilise une architecture de type Graph Neural Network. Ces travaux ont été effectués par Jules Caposiena, stagiaire à l'Institut Pascal. Ma contribution a été

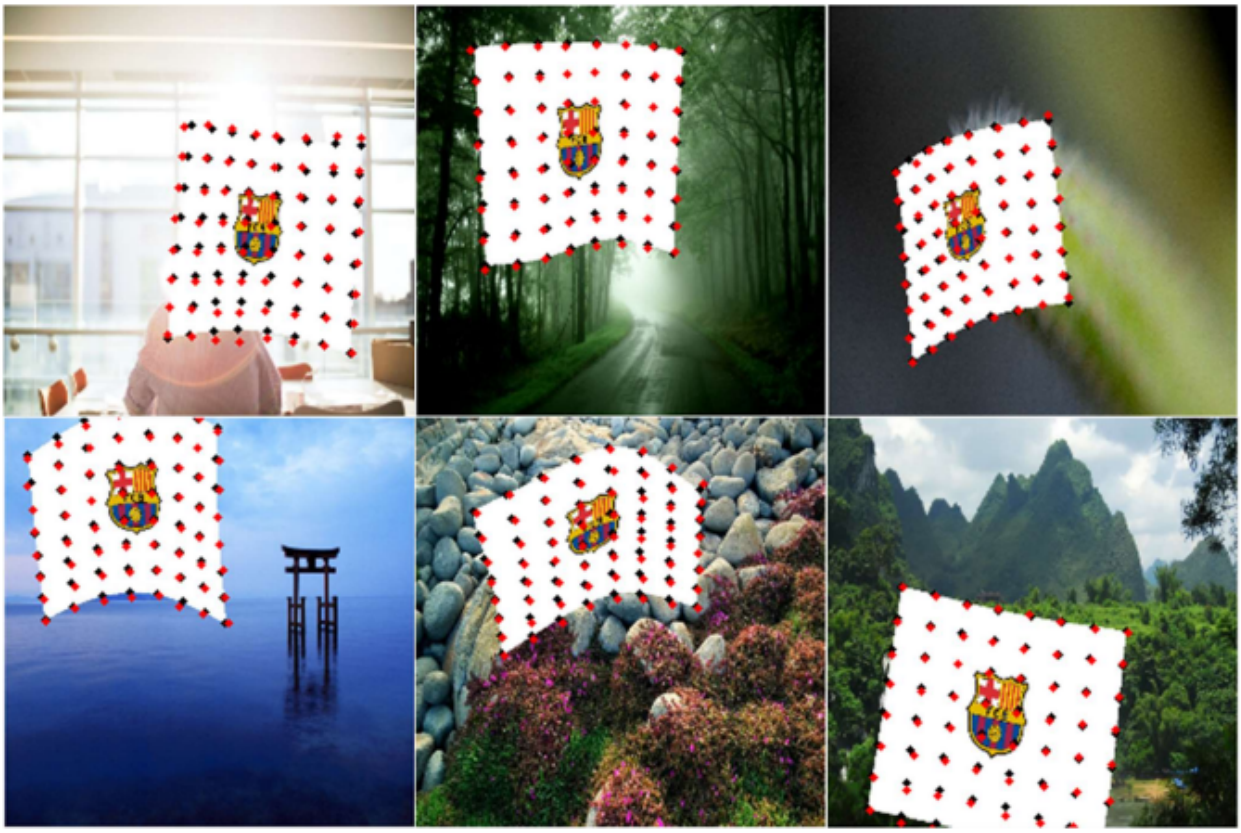

FIGURE C.2: Résultat de la méthode Graph Neural Network

Image issue des travaux et réalisée par Jules Caposiena, utilisée dans son rapport de stage

d'aider à la compréhension du cas d'application réel et ses contraintes, ainsi que le support aval d'intégration si la méthode aboutissait.
L'idée était de générer des données labellisée simulées sous Blender. Le rendu essayait d'être le plus réaliste possible, en particulier en proposant de nombreux éclairage pour une même scènes. Les données obtenues était utilisées pour de l'apprentissage supervisé. L'erreur était calculé au sens des moindres carré, afin de minimiser les distances entre référence et sortie du réseau.
A la fin du stage, le réseau de neurones arrivait à suivre l'objet avec uniquement les données monoculaires, illustré Fig. C.2. Cependant, la précision était trop basse – avec des erreurs de l'ordre du centimètre sur la position des points a détecter, ce qui fait que nous n'avons pas intégré ces travaux dans la solution finale.

Approches de modèles non exploitées

Deux approches prometteuses n'ont pas été exploitées.

— Meshless Shape Matching semblait adapté au cas de la gomme. Cependant, la nature sans maillage du modèle rend son implémentation complexe, en particulier pour exploiter des informations de perception.
— Nous pensions tirer profit de la géométrie de la gomme en modélisant les bandes de gommes par des ressort duaux-quaternion couplés par les sillons. Par manque de temps, nous n'avons pas pu nous pencher plus sur cette formulation du problème.

Annexe D

Bibliographie

IROS

[Agh+22] Omid AGHAJANZADEH, Miguel ARANDA, Gonzalo LÓPEZ-NICOLÁS, Roland LENAIN et Youcef MEZOUAR. “An offline geometric model for controlling the shape of elastic linear objects”. In : *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2022, p. 2175-2181.

[Avi+22] Yahav AVIGAL, Lars BERSCHEID, Tamim ASFOUR, Torsten KRÖGER et Ken GOLDBERG. “Speedfolding : Learning efficient bimanual folding of garments”. In : *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2022, p. 1-8.

[Bea+21] Samuel BEAUSSANT, Sébastien LENGAGNE, Benoit THUILOT et Olivier STASSE. “Delay aware universal notice network : real world multi-robot transfer learning”. In : *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2021, p. 1251-1258.

[Ber13] Dmitry BERENSON. “Manipulation of deformable objects without modeling and simulating deformation”. In : *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2013, p. 4525-4532.

[Bro+05] Alex BROOKS, Tobias KAUPP, Alexei MAKARENKO, Stefan WILLIAMS et Anders OREBACK. “Towards component-based robotics”. In : *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2005, p. 163-168.

[Den+22] Yuhong DENG, Chongkun XIA, Xueqian WANG et Lipeng CHEN. “Deep reinforcement learning based on local GNN for goal-conditioned deformable object rearranging”. In : *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2022, p. 1131-1138.

[Dru+22] Juliette DRUPT, Claire DUNE, Andrew I COMPORT, Sabine SEILLIER et Vincent HUGEL. “Inertial-measurement-based catenary shape estimation of underwater cables for tethered robots”. In : *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2022, p. 6867-6872.

[Due+18] Simon DUENSER, James M BERN, Roi PORANNE et Stelian COROS. “Interactive robotic manipulation of elastic objects”. In : *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2018, p. 3476-3481.

[Fil+22] N Roca FILELLA, Adrien KOESSLER, BC BOUZGARROU et J-A Corrales RAMON. "3D visual-based tension control in strip-like deformable objects using a catenary model". In : *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2022, p. 3210-3217.

[GS14] Mevlana C GEMICI et Ashutosh SAXENA. "Learning haptic representation for manipulating deformable food objects". In : *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 2014, p. 638-645.

[Gül+17] Püren GÜLER, Alessandro PIEROPAN, Masatoshi ISHIKAWA et Danica KRAGIC. "Estimating deformability of objects using meshless shape matching". In : *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2017, p. 5941-5948.

[Hie+22] Julius HIETALA, David BLANCO-MULERO, Gokhan ALCAN et Ville KYRKI. "Learning visual feedback control for dynamic cloth folding". In : *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2022, p. 1455-1462.

[KBS22] Azarakhsh KEIPOUR, Maryam BANDARI et Stefan SCHAAL. "Efficient spatial representation and routing of deformable one-dimensional objects for manipulation". In : *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2022, p. 211-216.

[Mar+21] Francisco MARTÍN, Jonatan Ginés CLAVERO, Vicente MATELLÁN et Francisco J RODRÍGUEZ. "Plansys2 : A planning system framework for ros2". In : *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2021, p. 9742-9749.

[Ouy+18] Bo OUYANG, Hangjie MO, Haoyao CHEN, Yunhui LIU et Dong SUN. "Robust model-predictive deformation control of a soft object by using a flexible continuum robot". In : *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2018, p. 613-618.

[PLS15] Antoine PETIT, Vincenzo LIPPIELLO et Bruno SICILIANO. "Real-time tracking of 3D elastic objects with an RGB-D sensor". In : *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2015, p. 3914-3921.

[SLK22] Finn SÜBERKRÜB, Rita LAEZZA et Yiannis KARAYIANNIDIS. "Feel the tension : Manipulation of deformable linear objects in environments with fixtures using force information". In : *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2022, p. 11216-11222.

[YSS22] Yuxuan YANG, Johannes A STORK et Todor STOYANOV. "Online Model Learning for Shape Control of Deformable Linear Objects". In : *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2022, p. 4056-4062.

ICRA

[CD17] Berk CALLI et Aaron M DOLLAR. "Vision-based model predictive control for within-hand precision manipulation with underactuated grippers". In : *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2017, p. 2839-2845.

[Her+22] Rafael HERGUEDAS, Miguel ARANDA, Gonzalo LÓPEZ-NICOLÁS, Carlos SAGÜÉS et Youcef MEZOUAR. "Multirobot control with double-integrator dynamics and control barrier functions for deformable object transport". In : *2022 International Conference on Robotics and Automation (ICRA)*. IEEE. 2022, p. 1485-1491.

[LKM20] Romain LAGNEAU, Alexandre KRUPA et Maud MARCHAL. "Active deformation through visual servoing of soft objects". In : *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2020, p. 8978-8984.

[Qui+09] Morgan QUIGLEY, Ken CONLEY, Brian GERKEY, Josh FAUST, Tully FOOTE, Jeremy LEIBS, Rob WHEELER, Andrew Y NG et al. "ROS : an open-source Robot Operating System". In : *ICRA workshop on open source software*. T. 3. 3.2. Kobe, Japan. 2009, p. 5.

[Shi+19] Changyeob SHIN, Peter Walker FERGUSON, Sahba Aghajani PEDRAM, Ji MA, Erik P DUTSON et Jacob ROSEN. "Autonomous tissue manipulation via surgical robot using learning based model predictive control". In : *2019 International Conference on Robotics and Automation (ICRA)*. IEEE. 2019, p. 3875-3881.

Autres conférences IEEE

[Gan+19] Juan M GANDARIAS, Francisco PASTOR, Alfonso J GARCIA-CEREZO et Jesús M Gómez-de GABRIEL. "Active tactile recognition of deformable objects with 3d convolutional neural networks". In : *2019 IEEE World Haptics Conference (WHC)*. IEEE. 2019, p. 551-555.

[Gul+15] Puren GULER, Karl PAUWELS, Alessandro PIEROPAN, Hedvig KJELLSTRÖM et Danica KRAGIC. "Estimating the deformability of elastic materials using optical flow and position-based dynamics". In : *2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids)*. IEEE. 2015, p. 965-971.

[He+16] Kaiming HE, Xiangyu ZHANG, Shaoqing REN et Jian SUN. "Deep residual learning for image recognition". In : *Proceedings of the IEEE conference on computer vision and pattern recognition*. IEEE. 2016, p. 770-778.

[He+17] Kaiming HE, Georgia GKIOXARI, Piotr DOLLÁR et Ross GIRSHICK. "Mask r-cnn". In : *Proceedings of the IEEE iInternational Conference on Computer Vision*. IEEE. 2017, p. 2961-2969.

[JC88] JOSIN et CHARNEY. "Robot control using neural networks". In : *IEEE 1988 International Conference on Neural Networks*. IEEE. 1988, p. 625-631.

[KP12] Steven KINIO et Alexandru PATRICIU. "A comparative study of H∞ and PID control for indirect deformable object manipulation". In : *2012 IEEE International Conference on Robotics and Biomimetics (ROBIO)*. IEEE. 2012, p. 414-420.

[Par+15] Shaifali PARASHAR, Daniel PIZARRO, Adrien BARTOLI et Toby COLLINS. "As-rigid-as-possible volumetric shape-from-template". In : *Proceedings of the IEEE International Conference on Computer Vision*. IEEE. 2015, p. 891-899.

[Sim+21] Mihael SIMONIČ, Timotej GAŠPAR, Saeed ABDOLSHAH, Sami HADDADIN, Manuel G CATALANO, Florentin WÖRGÖTTER, Aleš UDE et al. "Modular ROS-based software architecture for reconfigurable, Industry 4.0 compatible robotic workcells". In : *2021 20th International Conference on Advanced Robotics (ICAR)*. IEEE. 2021, p. 44-51.

[SKM19a] Agniva SENGUPTA, Alexandre KRUPA et Eric MARCHAND. “Tracking of non-rigid objects using rgb-d camera”. In : *2019 IEEE International Conference on Systems, Man and Cybernetics (SMC)*. IEEE. 2019, p. 3310-3317.

[SKM19b] Agniva SENGUPTA, Alexandre KRUPA et Eric MARCHAND. “Tracking of non-rigid objects using rgb-d camera”. In : *2019 IEEE International Conference on Systems, Man and Cybernetics (SMC)*. IEEE. 2019, p. 3310-3317.

[Tia+13] Yuan TIAN, Yin YANG, Xiaohu GUO et Balakrishnan PRABHAKARAN. “Haptic-enabled interactive rendering of deformable objects based on shape matching”. In : *2013 IEEE International Symposium on Haptic Audio Visual Environments and Games (HAVE)*. IEEE. 2013, p. 75-80.

[WLS12] Stefanie WUHRER, Jochen LANG et Chang SHU. “Tracking complete deformable objects with finite elements”. In : *2012 Second International Conference on 3D Imaging, Modeling, Processing, Visualization & Transmission*. IEEE. 2012, p. 1-8.

[Zak+22] Mélodie Hani Daniel ZAKARIA, Miguel ARANDA, Laurent LEQUIÈVRE, Sébastien LENGAGNE, Juan Antonio Corrales RAMÓN et Youcef MEZOUAR. “Robotic control of the deformation of soft linear objects using deep reinforcement learning”. In : *2022 IEEE 18th International Conference on Automation Science and Engineering (CASE)*. IEEE. 2022, p. 1516-1522.

Autres conférences

[BC00] David BOURGUIGNON et Marie-Paule CANI. “Controlling anisotropy in mass-spring systems”. In : *Computer Animation and Simulation 2000 : Proceedings of the Eurographics Workshop in Interlaken, Switzerland, August 21–22, 2000*. Springer. 2000, p. 113-123.

[Bia+04] Gérald BIANCHI, Barbara SOLENTHALER, Gábor SZÉKELY et Matthias HARDERS. “Simultaneous topology and stiffness identification for mass-spring models based on fem reference deformations”. In : *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2004 : 7th International Conference, Saint-Malo, France, September 26-29, 2004. Proceedings, Part II 7*. Springer. 2004, p. 293-301.

[BSP09] Jernej BARBIČ, Marco da SILVA et Jovan POPOVIĆ. “Deformable object animation using reduced optimal control”. In : *ACM SIGGRAPH 2009 papers*. 2009, p. 1-9.

[DSB99] Mathieu DESBRUN, Peter SCHRÖDER et Alan BARR. “Interactive animation of structured deformable objects”. In : *Graphics Interface*. T. 99. 5. 1999, p. 10.

[Kav+07] Ladislav KAVAN, Steven COLLINS, Jiří ŽÁRA et Carol O’SULLIVAN. “Skinning with dual quaternions”. In : *Proceedings of the 2007 symposium on Interactive 3D graphics and games*. 2007, p. 39-46.

[MMC16] Miles MACKLIN, Matthias MÜLLER et Nuttapong CHENTANEZ. “XPBD : position-based simulation of compliant constrained dynamics”. In : *Proceedings of the 9th International Conference on Motion in Games*. 2016, p. 49-54.

[Pro+95] Xavier PROVOT et al. “Deformation constraints in a mass-spring model to describe rigid cloth behaviour”. In : *Graphics interface*. Canadian Information Processing Society. 1995, p. 147-147.

[RFB15] Olaf RONNEBERGER, Philipp FISCHER et Thomas BROX. "U-net : Convolutional networks for biomedical image segmentation". In : *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015 : 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18*. Springer. 2015, p. 234-241.

[SA07] Olga SORKINE et Marc ALEXA. "As-rigid-as-possible surface modeling". In : *Symposium on Geometry processing*. T. 4. Citeseer. 2007, p. 109-116.

[Ter+87] Demetri TERZOPOULOS, John PLATT, Alan BARR et Kurt FLEISCHER. "Elastically deformable models". In : *Proceedings of the 14th annual conference on Computer graphics and interactive techniques*. 1987, p. 205-214.

[TF88] Demetri TERZOPOULOS et Kurt FLEISCHER. "Modeling inelastic deformation : viscolelasticity, plasticity, fracture". In : *Proceedings of the 15th annual conference on Computer graphics and interactive techniques*. 1988, p. 269-278.

[Wu+20a] Yilin WU, Wilson YAN, Thanard KURUTACH, Lerrel PINTO et Pieter ABBEEL. "Learning to Manipulate Deformable Objects without Demonstrations". In : *16th Robotics : Science and Systems, RSS 2020*. MIT Press Journals. 2020.

RAL

[Ala+18] Farshid ALAMBEIGI, Zerui WANG, Rachel HEGEMAN, Yun-Hui LIU et Mehran ARMAND. "A robust data-driven approach for online learning and manipulation of unmodeled 3-D heterogeneous compliant objects". In : *IEEE Robotics and Automation Letters* 3.4 (2018), p. 4140-4147.

[BWV22] Harish BEZAWADA, Cole WOODS et Vishesh VIKAS. "Shape Reconstruction of Soft Manipulators Using Vision and IMU Feedback". In : *IEEE Robotics and Automation Letters* 7.4 (2022), p. 9589-9596.

[Cap+22] Alessio CAPORALI, Kevin GALASSI, Riccardo ZANELLA et Gianluca PALLI. "FASTDLO : Fast deformable linear objects instance segmentation". In : *IEEE Robotics and Automation Letters* 7.4 (2022), p. 9075-9082.

[CH06] François CHAUMETTE et Seth HUTCHINSON. "Visual servo control. I. Basic approaches". In : *IEEE Robotics & Automation Magazine* 13.4 (2006), p. 82-90.

[CH07] François CHAUMETTE et Seth HUTCHINSON. "Visual servo control. II. Advanced approaches [Tutorial]". In : *IEEE Robotics & Automation Magazine* 14.1 (2007), p. 109-118.

[SB+22] Mohammadreza SHETAB-BUSHEHRI, Miguel ARANDA, Youcef MEZOUAR et Erol ÖZGÜR. "As-rigid-as-possible shape servoing". In : *IEEE Robotics and Automation Letters* 7.2 (2022), p. 3898-3905.

[Yan+20] Mengyuan YAN, Yilin ZHU, Ning JIN et Jeannette BOHG. "Self-supervised learning of state estimation for manipulating deformable linear objects". In : *IEEE robotics and automation letters* 5.2 (2020), p. 2372-2379.

[ZPC21] Simon ZIMMERMANN, Roi PORANNE et Stelian COROS. "Dynamic manipulation of deformable objects with implicit integration". In : *IEEE Robotics and Automation Letters* 6.2 (2021), p. 4209-4216.

Autres journaux IEEE

[AMN17] Antonio AGUDO et Francesc MORENO-NOGUER. “Force-based representation for non-rigid shape and elastic model estimation”. In : *IEEE Transactions on pattern analysis and machine intelligence* 40.9 (2017), p. 2137-2150.

[Bar+15] Adrien BARTOLI, Yan GÉRARD, Francois CHADEBECQ, Toby COLLINS et Daniel PIZARRO. “Shape-from-template”. In : *IEEE Transactions on pattern analysis and machine intelligence* 37.10 (2015), p. 2099-2118.

[Ben96] Stuart BENNETT. “A brief history of automatic control”. In : *IEEE Control Systems Magazine* 16.3 (1996), p. 17-25.

[Ebe+19] Donald EBEIGBE, Thang NGUYEN, Hanz RICHTER et Dan SIMON. “Robust regressor-free control of rigid robots using function approximations”. In : *IEEE Transactions on Control Systems Technology* 28.4 (2019), p. 1433-1446.

[Fis+19] Harrisson FISCHER, Margot VULLIEZ, Pierre LAGUILLAUMIE, Philippe VULLIEZ et Jean-Pierre GAZEAU. “RTRobMultiAxisControl : A framework for real-time multiaxis and multirobot control”. In : *IEEE Transactions on Automation Science and Engineering* 16.3 (2019), p. 1205-1217.

[Han+21] Lijun HAN, Hesheng WANG, Zhe LIU, Weidong CHEN et Xiufeng ZHANG. “Visual tracking control of deformable objects with a FAT-based controller”. In : *IEEE Transactions on Industrial Electronics* 69.2 (2021), p. 1673-1681.

[Har06] Warren HARRISON. “Eating your own dog food”. In : *IEEE Software* 23.3 (2006), p. 5-7.

[LG12] Zohar LEVI et Craig GOTSMAN. “D-Snake : image registration by as-similar-as-possible template deformation”. In : *IEEE Transactions on Visualization and Computer Graphics* 19.2 (2012), p. 331-343.

[LMV92] Antonio M LEPSCHY, Gian Antonio MIAN et Umberto VIARO. “Feedback control in ancient water and mechanical clocks”. In : *IEEE Transactions on Education* 35.1 (1992), p. 3-10.

[LSH07] Bryn LLOYD, Gábor SZÉKELY et Matthias HARDERS. “Identification of spring parameters for deformable object simulation”. In : *IEEE Transactions on Visualization and Computer Graphics* 13.5 (2007), p. 1081-1094.

[MTK96] Kim-Fung MAN, Kit-Sang TANG et Sam KWONG. “Genetic algorithms : concepts and applications [in engineering design]”. In : *IEEE Transactions on Industrial Electronics* 43.5 (1996), p. 519-534.

[NAL17] David NAVARRO-ALARCON et Yun-Hui LIU. “Fourier-based shape servoing : A new feedback method to actively deform soft objects into desired 2-D image contours”. In : *IEEE Transactions on Robotics* 34.1 (2017), p. 272-279.

[Nam+20] Changjoo NAM, Seokjun LEE, Jeongho LEE, Sang Hun CHEONG, Dong Hwan KIM, Changhwan KIM, Incheol KIM et Sung-Kee PARK. “A software architecture for service robots manipulating objects in human environments”. In : *IEEE Access* 8 (2020), p. 117900-117920.

[Whi69] Daniel E WHITNEY. “Resolved motion rate control of manipulators and human prostheses”. In : *IEEE Transactions on man-machine systems* 10.2 (1969), p. 47-53.

[Wu+20b] Zonghan WU, Shirui PAN, Fengwen CHEN, Guodong LONG, Chengqi ZHANG et S Yu PHILIP. "A comprehensive survey on graph neural networks". In : *IEEE Transactions on neural networks and learning systems* 32.1 (2020), p. 4-24.

[Yan+18] Mingkai YANG, Yanling LI, Hao DU, Chen LI et Zhengyou HE. "Hierarchical multiobjective H-infinity robust control design for wireless power transfer system using genetic algorithm". In : *IEEE Transactions on Control Systems Technology* 27.4 (2018), p. 1753-1761.

International Journal of Robotics Research

[BM14] Timothy BRETL et Zoe MCCARTHY. "Quasi-static manipulation of a Kirchhoff elastic rod based on a geometric analysis of equilibrium configurations". In : *The International Journal of Robotics Research* 33.1 (2014), p. 48-68.

[Che+20] Andrea CHERUBINI, Valerio ORTENZI, Akansel COSGUN, Robert LEE et Peter CORKE. "Model-free vision-based shaping of deformable plastic materials". In : *The International Journal of Robotics Research* 39.14 (2020), p. 1739-1759.

[San+18] Jose SANCHEZ, Juan-Antonio CORRALES, Belhassen-Chedli BOUZGARROU et Youcef MEZOUAR. "Robotic manipulation and sensing of deformable objects in domestic and industrial applications : a survey". In : *The International Journal of Robotics Research* 37.7 (2018), p. 688-716.

[TT22] Te TANG et Masayoshi TOMIZUKA. "Track deformable objects from point clouds with structure preserved registration". In : *The International Journal of Robotics Research* 41.6 (2022), p. 599-614.

[Vro+17] Gustaaf J VROOIJINK, Alper DENASI, Jan G GRANDJEAN et Sarthak MISRA. "Model predictive control of a robotically actuated delivery sheath for beating heart compensation". In : *The International Journal of Robotics Research* 36.2 (2017), p. 193-209.

Autres journaux

[AB16] Aakash AHMAD et Muhammad Ali BABAR. "Software architectures for robotic systems : A systematic mapping study". In : *Journal of Systems and Software* 122 (2016), p. 16-39.

[Aba+17] Hernán ABAUNZA, Pedro CASTILLO, A VICTORINO et Rogelio LOZANO. "Dual quaternion modeling and control of a quad-rotor aerial manipulator". In : *Journal of Intelligent & Robotic Systems* 88 (2017), p. 267-283.

[ABPZ21] Meghna P AYYAR, Jenny BENOIS-PINEAU et Akka ZEMMARI. "Review of white box methods for explanations of convolutional neural networks in image classification tasks". In : *Journal of Electronic Imaging* 30.5 (2021), p. 050901-050901.

[Arg+18] Albert ARGILAGA, Jacquees DESRUES, Stefano DAL PONT, Gaël COMBE et Denis CAILLERIE. "FEM× DEM multiscale modeling : Model performance enhancement from Newton strategy to element loop parallelization". In : *International Journal for Numerical Methods in Engineering* 114.1 (2018), p. 47-65.

[Cli60] Truesdell CLIFFORD. "The Rational Mechanics of Flexible or Elastic Bodies : 1638-1788". In : *Leonhardi Euleri Opera Omnia, Ser. 2* (1960).

[Dua06] Yingxuan DUAN. “Fundamental tradeoff between performance and robustness in control design : Yingxuan Duan.” In : (2006).

[FJ+22] David FUENTES-JIMENEZ, Daniel PIZARRO, David CASILLAS-PÉREZ, Toby COLLINS et Adrien BARTOLI. “Deep Shape-from-Template : Single-image quasi-isometric deformable registration and reconstruction”. In : *Image and Vision Computing* 127 (2022), p. 104531.

[For96] Stephanie FORREST. “Genetic algorithms”. In : *ACM computing surveys (CSUR)* 28.1 (1996), p. 77-80.

[FY97] Brian FOOTE et Joseph YODER. “Big ball of mud”. In : *Pattern languages of program design* 4 (1997), p. 654-692.

[Gu+18] Jiuxiang GU et al. “Recent advances in convolutional neural networks”. In : *Pattern recognition* 77 (2018), p. 354-377.

[Han+18] Tao HAN, Xuan ZHAO, Peigen SUN et Jia PAN. “Robust shape estimation for 3D deformable object manipulation”. In : *Communications in Information and Systems* 18.2 (2018), p. 107-124.

[Hoc98] Sepp HOCHREITER. “The vanishing gradient problem during learning recurrent neural nets and problem solutions”. In : *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems* 6.02 (1998), p. 107-116.

[HWK19] Phillip HYATT, David WINGATE et Marc D KILLPACK. “Model-based control of soft actuators using learned non-linear discrete-time models”. In : *Frontiers in Robotics and AI* 6 (2019), p. 22.

[II09] Takeo IGARASHI et Yuki IGARASHI. “Implementing as-rigid-as-possible shape manipulation and surface flattening”. In : *journal of graphics, gpu, and game tools* 14.1 (2009), p. 17-30.

[Jia+17] Tao JIANG, Kun QIAN, Shuang LIU, Jing WANG, Xiaosong YANG et Jianjun ZHANG. “Consistent as-similar-as-possible non-isometric surface registration”. In : *The Visual Computer* 33 (2017), p. 891-901.

[Kav+06a] Ladislav KAVAN, Steven COLLINS, Carol O’SULLIVAN et Jiri ZARA. “Dual quaternions for rigid transformation blending”. In : *Trinity College Dublin* 5 (2006).

[Kav+06b] Ladislav KAVAN, Steven COLLINS, Carol O’SULLIVAN et Jiri ZARA. “Dual quaternions for rigid transformation blending”. In : *Trinity College Dublin* 5 (2006).

[KLM96] Leslie Pack KAELBLING, Michael L LITTMAN et Andrew W MOORE. “Reinforcement learning : A survey”. In : *Journal of artificial intelligence research* 4 (1996), p. 237-285.

[Lya92] Aleksandr Mikhailovich LYAPUNOV. “The general problem of the stability of motion”. In : *International journal of control* 55.3 (1992), p. 531-534.

[Mac+14] Miles MACKLIN, Matthias MÜLLER, Nuttapong CHENTANEZ et Tae-Yong KIM. “Unified particle physics for real-time applications”. In : *ACM Transactions on Graphics (TOG)* 33.4 (2014), p. 1-12.

[Mae11] Hamid Reza MAEI. “Gradient temporal-difference learning algorithms”. In : (2011).

[Mar00] Robert C MARTIN. “Design principles and design patterns”. In : *Object Mentor* 1.34 (2000), p. 597.

[Max68] James Clerk MAXWELL. “I. On governors”. In : *Proceedings of the Royal Society of London* 16 (1868), p. 270-283.

[MP43] Warren S MCCULLOCH et Walter PITTS. “A logical calculus of the ideas immanent in nervous activity”. In : *The bulletin of mathematical biophysics* 5 (1943), p. 115-133.

[Mül+05] Matthias MÜLLER, Bruno HEIDELBERGER, Matthias TESCHNER et Markus GROSS. “Meshless deformations based on shape matching”. In : *ACM transactions on graphics (TOG)* 24.3 (2005), p. 471-478.

[Mül+07] Matthias MÜLLER, Bruno HEIDELBERGER, Marcus HENNIX et John RATCLIFF. “Position based dynamics”. In : *Journal of Visual Communication and Image Representation* 18.2 (2007), p. 109-118.

[Mül08] Matthias MÜLLER. “Hierarchical position based dynamics”. In : (2008).

[NVP18] Félix NADON, Angel J VALENCIA et Pierre PAYEUR. “Multi-modal sensing and robotic manipulation of non-rigid objects : A survey”. In : *Robotics* 7.4 (2018), p. 74.

[ÖB17] Erol ÖZGÜR et Adrien BARTOLI. “Particle-SfT : A provably-convergent, fast shape-from-template algorithm”. In : *International Journal of Computer Vision* 123 (2017), p. 184-205.

[Pan05] Olivier PANTALÉ. “Parallelization of an object-oriented FEM dynamics code : influence of the strategies on the Speedup”. In : *Advances in Engineering Software* 36.6 (2005), p. 361-373.

[Pan+15] Junjun PAN, Junxuan BAI, Xin ZHAO, Aimin HAO et Hong QIN. “Real-time haptic manipulation and cutting of hybrid soft tissue models by extended position-based dynamics”. In : *Computer Animation and Virtual Worlds* 26.3-4 (2015), p. 321-335.

[Rov+17] Francesco ROVIDA, Matthew CROSBY, Dirk HOLZ, Athanasios S POLYDOROS, Bjarne GROSSMANN, Ronald PA PETRICK et Volker KRÜGER. “SkiROS—a skill-based robot control platform on top of ROS”. In : *Robot Operating System (ROS) The Complete Reference (Volume 2)* (2017), p. 121-160.

[Sch+14] Christian SCHULZ, Christoph von TYCOWICZ, Hans-Peter SEIDEL et Klaus HILDEBRANDT. “Animating deformable objects using sparse spacetime constraints”. In : *ACM Transactions on Graphics (TOG)* 33.4 (2014), p. 1-10.

[SO+20] David ST-ONGE, Vivek Shankar VARADHARAJAN, Ivan ŠVOGOR et Giovanni BELTRAME. “From design to deployment : Decentralized coordination of heterogeneous robotic teams”. In : *Frontiers in Robotics and AI* 7 (2020), p. 51.

[Sta10] Ioannis STAMELOS. “Software project management anti-patterns”. In : *Journal of Systems and Software* 83.1 (2010), p. 52-59.

[SV+09] Gaizka SAN VICENTE, Carlos BUCHART, Diego BORRO et Juan Tomas CELIGÜETA. “Maxillofacial surgery simulation using a mass-spring model derived from continuum and the scaled displacement method”. In : *International journal of computer assisted radiology and surgery* 4 (2009), p. 89-98.

[TS09] Lisa TORREY et Jude SHAVLIK. “Transfer Learning”. In : *Handbook of Research on Machine Learning Applications and Trends : Algorithms, Methods, and Techniques : Algorithms, Methods, and Techniques* (2009), p. 242.

[WY10] Weina WU et Zhong YOU. “Modelling rigid origami with quaternions and dual quaternions”. In : *Proceedings of the Royal Society A : Mathematical, physical and engineering sciences* 466.2119 (2010), p. 2155-2174.

[Zho+20] Jie ZHOU et al. “Graph neural networks : A review of methods and applications”. In : *AI open* 1 (2020), p. 57-81.

[ZN42] John G ZIEGLER et Nathaniel B NICHOLS. “Optimum settings for automatic controllers”. In : *Transactions of the American society of mechanical engineers* 64.8 (1942), p. 759-765.

Livres

[AM07] Brian DO ANDERSON et John B MOORE. *Optimal control : linear quadratic methods*. Courier Corporation, 2007.

[BB08] Tamer BAŞAR et Pierre BERNHARD. *H-infinity optimal control and related minimax design problems : a dynamic game approach*. Springer Science & Business Media, 2008.

[Bou+14] Yiannis BOUTALIS, Dimitrios THEODORIDIS, Theodore KOTTAS et Manolis A CHRISTODOULOU. *System identification and adaptive control. Theory and Applications of the Neurofuzzy and Fuzzy Cognitive Network Models.* Springer, 2014.

[Cam+07] Eduardo F CAMACHO, Carlos BORDONS, Eduardo F CAMACHO et Carlos BORDONS. *Model predictive controllers*. Springer, 2007.

[CC08] Matthieu CORD et Pádraig CUNNINGHAM. *Machine learning techniques for multimedia : case studies on organization and retrieval*. Springer Science & Business Media, 2008.

[CC12] Charles K CHUI et Guanrong CHEN. *Linear systems and optimal control*. T. 18. Springer Science & Business Media, 2012.

[Cro+05] James CROWE et al. *PID control : new identification and design methods*. Springer, 2005.

[Gam+95] Erich GAMMA, Richard HELM, Ralph JOHNSON et John VLISSIDES. *Design patterns : elements of reusable object-oriented software*. Pearson Deutschland GmbH, 1995.

[GBC16] Ian GOODFELLOW, Yoshua BENGIO et Aaron COURVILLE. *Deep learning*. MIT press, 2016.

[Gov19] Felix GOVAERS. *Introduction and implementations of the Kalman filter*. BoD–Books on Demand, 2019.

[LB13] Mats G LARSON et Fredrik BENGZON. *The finite element method : theory, implementation, and applications*. T. 10. Springer Science & Business Media, 2013.

[LC19] Yann LE CUN. *Quand la machine apprend : la révolution des neurones artificiels et de l'apprentissage profond*. Odile Jacob, 2019.

[LLM+98] Ioan Doré LANDAU, Rogelio LOZANO, Mohammed M'SAAD et al. *Adaptive control*. T. 51. Springer New York, 1998.

[LS12] Joshua D LAMOS-SWEENEY. *Deep learning using genetic algorithms*. Rochester Institute of Technology, 2012.

[Nak90] Yoshihiko NAKAMURA. *Advanced robotics : redundancy and optimization*. Addison-Wesley Longman Publishing Co., Inc., 1990.

[New33] Isaac NEWTON. *Philosophiae naturalis principia mathematica*. T. 1. G. Brookman, 1833.

Brevets

[Hou62] Paul VC HOUGH. *Method and means for recognizing complex patterns*. US Patent 3,069,654. 1962.

Publications issus de la thèse

[Cha+23] Rohit CHANDRA, Victor H GIRAUD, Mohammad ALKHATIB et Youcef MEZOUAR. "Dual quaternion based dynamic movement primitives to learn industrial tasks using teleoperation". In : *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2023, p. 3757-3763.

[Gir+22] Victor H GIRAUD, Maxime PADRIN, Mohammadreza SHETAB-BUSHEHRI, Chedli BOUZGARROU, Youcef MEZOUAR et Erol OZGUR. "Optimal Shape Servoing with Task-focused Convergence Constraints". In : *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2022, p. 2197-2202.

[Gir+23] Victor H GIRAUD, Mohammadreza SHETAB-BUSHEHRI, Nicolas ROCA-FILELLA et Jean-Marie. DETTORRE. *Système Robotique Bi-Bras de confection de pneumatiques*. 2023.