



HAL
open science

Learning by Demonstration for Grasp Planning: application to Pluri-digital and Underactuated Robotic Grasping

Clément Rolinat

► **To cite this version:**

Clément Rolinat. Learning by Demonstration for Grasp Planning: application to Pluri-digital and Underactuated Robotic Grasping. Signal and Image processing. Université Grenoble Alpes [2020-..], 2022. English. NNT: 2022GRALT044 . tel-03850676

HAL Id: tel-03850676

<https://theses.hal.science/tel-03850676>

Submitted on 14 Nov 2022

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Thèse

pour obtenir le grade de

Docteur de l'Université Grenoble Alpes

Spécialité : Signal, Image, Parole, Télécoms

Présentée par

Clément ROLINAT

Thèse dirigée par :

Dr. Christelle GODIN, Directrice de recherche au CEA Leti, HDR
et co-encadrée par :

Dr. Mathieu GROSSARD, Directeur de recherche au CEA List, HDR

Dr. Saïfeddine ALOUI, Ingénieur de recherche au CEA Leti

Préparée au sein du service de robotique interactive du **CEA List**
dans l'École doctorale **Electronique, Electrotechnique,**
Automatique et Traitement du Signal (EEATS)

Learning by Demonstration for Grasp Planning - Application to Pluri-digital and Underactuated Robotic Grasping

Apprentissage par démonstration pour la planification de prises - application à la préhension robotique pluri-digitale et sous-actionnée

Thèse soutenue publiquement le **15 juin 2022**, devant le jury composé de :

Pr. Philippe BIDAUD

Professeur à Sorbonne Université, président du jury

Pr. Youcef MEZOUAR

Professeur à Clermont Auvergne INP – SIGMA'Clermont, rapporteur

Dr. Jean-Pierre GAZEAU

Ingénieur de Recherche à l'institut Pprime du CNRS, HDR, rapporteur

Pr. Pedro RODRIGUEZ-AYERBE

Professeur à CentraleSupélec, Université Paris-Saclay, examinateur

Dr. Sylvie CHARBONNIER

Maître de Conférences à l'Université Grenoble Alpes, HDR, examinatrice

Dr. Christelle GODIN

Directrice de recherche au CEA Leti, HDR, directrice de thèse



Remerciements

Je souhaite tout d'abord remercier vivement ma directrice de thèse, Christelle Godin, ainsi que mes encadrants, Mathieu Grossard et Saifeddine Aloui, pour leurs conseils pertinents, leurs critiques toujours constructives et bienveillantes, et plus généralement la grande qualité des nombreux échanges que j'ai pu avoir avec eux. Je les remercie également pour leur soutien et leur disponibilité tout au long de ma thèse.

J'exprime toute ma gratitude à Messieurs Jean-Pierre Gazeau, Ingénieur de Recherche HDR à l'institut Pprime du CNRS, et Youcef Mezouar, Professeur à Clermont Auvergne INP – SIGMA'Clermont, pour m'avoir fait l'honneur de rapporter ces travaux.

Je tiens également à remercier Monsieur Philippe Bidaud, Professeur à Sorbonne Université, de m'avoir fait l'honneur de présider ma soutenance de thèse. Je remercie aussi Madame Sylvie Charbonnier, Maître de Conférences HDR à l'Université Grenoble Alpes, et Monsieur Pedro Rodriguez-Ayerbe, Professeur à CentraleSupélec, d'avoir accepté d'évaluer mes travaux.

Je remercie ensuite Yann Perrot, directeur du service de robotique interactive au CEA List, et François Lansade, directeur du laboratoire d'architecture des systèmes robotiques au CEA List, pour m'avoir accueilli au sein de leur service et laboratoire et m'avoir offert de très bonnes conditions de travail.

Mes remerciements vont également à tous les collègues du service de robotique interactive pour la bonne ambiance qui y règne. Je tiens à remercier tout particulièrement les collègues du CEA qui ont contribué d'une façon ou d'une autre à ce travail de thèse, en particulier lors de la phase d'expérimentations. Je remercie aussi les collègues de la société Tridimeo pour le prêt d'une de leur caméra et leur aide pour son installation et utilisation.

Je remercie enfin chaleureusement mes amis et ma famille pour m'avoir offert des moments de décompressions bienvenus, ainsi que pour leur soutien sans faille et leurs encouragements durant mes années de thèse.

Contents

Résumé en français	7
Introduction	13
Context	13
Outlines & Contributions	16
1 Robotic Grasping: Principle and Techniques	19
1.1 Grasping in Industry: Overview	20
1.2 Toward More Versatile Grippers	29
1.3 Grasp Planning Principles	39
1.4 Grasp Space Exploration Issue	45
1.5 Conclusion	51
2 Grasp Modeling Framework	53
2.1 Grasp and Gripper Description	54
2.2 Grasp Characterisation	70
2.3 Conclusion	88
3 Human Initiated Grasp Space Exploration Approach	89
3.1 Problem Statement	90
3.2 Techniques for Data Generation & Dimensionality Reduction	95
3.3 Variational Auto-Encoder: Principle and Techniques	107
3.4 Variational Auto-Encoders for Grasp Space Exploration	114
3.5 Conclusion	120
4 Method Qualification	121
4.1 Considered Setup, Objects and Primitives Grasps	122
4.2 Hyperparameters Tuning and Latent Space Analysis	129
4.3 Experiments	137
4.4 Conclusion	144
5 Extension Towards Object Geometry Information	145
5.1 Approach	146
5.2 Learning to Compress Object Geometry	159

CONTENTS

5.3 Using Object Geometry to Generate Grasps	165
5.4 Conclusion	171
Conclusions & Perspectives	173
A Details of the Experimental Setup	181
Nomenclature	185
Acronyms	189
List of Figures	191
List of Tables	199
Bibliography	201

Résumé

La préhension et la manipulation sont des composantes essentielles de la robotique : le préhenseur est à l'interface entre le robot et l'objet d'intérêt. La manipulation nécessite une capacité de préhension, et fait référence à la réorientation et au repositionnement d'un objet saisi par rapport à un repère de référence donné. **Les capacités de préhension et de manipulation sont nécessaires dans la plupart des processus de fabrication industrielle**, car ils impliquent souvent, à un stade ou à un autre, des tâches de *pick-and-place*, d'assemblage ou de dévissage. **Les robots sont principalement introduits dans le processus de fabrication pour éviter les accidents du travail, le travail monotone et la charge mentale des opérateurs.** Plus précisément, certaines applications possibles de la préhension robotique dans l'industrie manufacturière sont par exemple : l'approvisionnement de machines-outils à commande numérique par ordinateur (CNC) pour une utilisation sans personnel, la palettisation, le levage et la manutention d'objets pour des raisons d'ergonomie, de propreté (pour les industries alimentaires, pharmaceutiques ou des semi-conducteurs) ou de sécurité [1].

Historiquement, dans l'industrie manufacturière, **les robots ont été développés pour remplacer les humains pour des tâches simples, répétitives et difficiles.** Les robots ont d'abord été introduits dans ce type de tâches car ce sont généralement les plus faciles à automatiser, celles qui présentent le plus de risques pour la sécurité et la santé de l'opérateur, et celles qui bénéficient le plus de l'augmentation de cadence permise par la substitution de l'humain par le système robotique. Dans ce contexte, **les robots ont été spécialisés en fonction de la tâche.** Cette stratégie de spécialisation a été utilisée dans de multiples composants et aspects du système robotique. **Elle est également visible pour les robots utilisés dans les tâches de préhension et de manipulation, dans le choix et la conception de l'architecture mécanique des préhenseurs et des logiciels associés.**

En effet, les préhenseurs robotiques et les algorithmes de planification de prises sont conçus spécifiquement pour la tâche cible, compte tenu des propriétés physiques de l'objet et des incertitudes concernant sa localisation et sa géométrie [1, 2]. **Les robots sont placés dans un environnement hautement contrôlé, et l'exécution correcte de la tâche de préhension repose sur des hypothèses fortes concernant l'objet à saisir.** Un changement dans les propriétés physiques de l'objet ou dans la disposition spatiale de la cellule robotique peut nécessiter des modifications de l'architecture du préhenseur ou de son planificateur de prises par un expert humain. Il s'agit d'un inconvénient

important qui empêche une utilisation plus large des robots dans les tâches industrielles nécessitant des capacités de préhension ou de manipulation.

Récemment, l'industrie 5.0 tend à remettre en question cette stratégie de spécialisation. En effet, l'objectif de ce concept est de promouvoir une industrie plus durable, plus centrée sur l'humain (à la fois les utilisateurs finaux et les opérateurs) et plus résiliente [3, 4]. En particulier, elle met en avant le concept de personnalisation de masse, qui vise à adapter chaque produit aux besoins de chaque client. Elle place également l'opérateur humain au centre de la chaîne de valeur, en utilisant, entre autres, des cobots (robots collaboratifs) conçus pour travailler de manière transparente en coopération avec un humain [5]. Pour atteindre les objectifs de l'industrie 5.0, une approche plus versatile de la robotique est nécessaire. **Ce besoin de versatilité (Définition 1) est également présent dans le cas particulier des préhenseurs robotiques et des algorithmes de planification de prises :**

- D'une part, les préhenseurs actuels ont du mal à gérer la grande variabilité des propriétés physiques de l'objet (telles que sa forme) ou de son attitude par rapport à un repère donné. Lorsque ce type de versatilité est requis, la capacité de préhension et la dextérité (Définition 2) de l'humain restent inégales. C'est souvent le cas lorsqu'un niveau élevé de personnalisation est nécessaire dans le produit. Cependant, en fonction de la tâche et de la nature de l'objet, cela peut entraîner certains risques corporels pour l'opérateur humain, tels qu'une probabilité accrue de développer des troubles musculo-squelettiques ou un taux plus élevé d'accidents du travail. Pour atténuer ce problème, il est possible d'utiliser des cobots. **Pour assister plus efficacement et de manière plus transparente l'opérateur humain lors d'une tâche de préhension qui nécessite une manipulation fine de l'objet, il faut des préhenseurs versatiles et des stratégies de planification de prises dont les capacités de manipulation sont proches de celles de la main humaine.**
- D'autre part, pour chaque tâche, le préhenseur correspondant doit être choisi parmi les préhenseurs existants, ou conçu de toutes pièces, de même que l'algorithme de planification de prises qui lui est associé. Cela peut représenter un coût important lors du développement d'une ligne de production pour un nouveau produit. **Posséder des préhenseurs polyvalents pourrait aider à réduire le temps et le coût d'intégration lors de la création de nouvelles lignes de fabrication ou de la réaffectation de lignes existantes** [6]. Comme pour le matériel, un besoin majeur est également d'apporter plus de versatilité dans la stratégie de planification de prises. Ces améliorations permettent de s'adapter plus facilement aux perturbations de la chaîne d'approvisionnement ou aux changements des besoins des clients.

Pour obtenir un robot doté de capacités de préhension versatiles, l'architecture mécanique du préhenseur doit être soigneusement conçue. **Il doit offrir un nombre suffisant de degrés de liberté pour produire une grande variété de prises pour**

diverses géométries d'objets. Dans ce contexte, une architecture pluridigitale est souvent choisie. Cependant, **l'architecture doit rester aussi simple que possible pour des contraintes de coût et de complexité du contrôleur.** Un moyen possible d'y parvenir est d'introduire un sous-actionnement (Définition 3) dans certaines parties de l'architecture mécanique du préhenseur. **La capacité de préhension du robot est également déterminée par son algorithme de planification de prises, qui dépend lui-même fortement de l'architecture mécanique choisie.** En effet, le but de l'algorithme de planification de prises est de choisir parmi différentes prises possibles, et de décider comment un objet donné doit être saisi dans une situation donnée. Les prises disponibles dépendent à la fois de l'objet et de l'architecture du préhenseur, ainsi que des contraintes de l'environnement ou de la tâche. **Pour choisir une prise appropriée à la situation parmi toutes les prises possibles, l'espace des prises (Définition 4) doit être exploré.**



Définition 1 (Versatilité). Dans le cadre de la préhension robotique, définit la capacité du préhenseur à réaliser une grande variété de tâches différentes, et à s'adapter à diverses géométries d'objets. Cette définition est dérivée de celle donnée dans GAZEAU [2].

Définition 2 (Dextérité). Capacité à déplacer un objet saisi par rapport au repère du préhenseur ou de la main selon une trajectoire donnée, tout en maintenant la stabilité de l'objet. Cette définition provient de GAZEAU [2].

Définition 3 (Sous-actionnement). Propriété d'un système d'avoir un vecteur d'entrée de plus petite dimension que le vecteur de sortie. Dans un contexte robotique, cela signifie avoir moins d'actionneurs que de degrés de liberté. Cette définition est donnée par BIRGLEN et al. [7].

Définition 4 (Espace des prises). Dans ces travaux, ce concept fait référence à l'ensemble de toutes les prises possibles qu'un préhenseur donné peut produire sur un objet donné. Cet espace est un sous-ensemble de l'ensemble des configurations du préhenseur.

Cette thèse rassemble plusieurs contributions publiées qui conduisent à la proposition d'un outil logiciel pour **explorer l'espace des prises d'un préhenseur robotique, particulièrement efficace dans le cas d'architectures pluridigitales et sous-actionnées.** Il utilise des Auto-Encodeurs Variationnels (VAE) pour générer des prises, ainsi qu'une prédiction de leur qualité, en s'inspirant de prises primitives spécifiées par démonstration par un opérateur.

La structure de ce manuscrit est divisée comme suit :

- **chapitre 1** : Les architectures de préhenseur et les algorithmes de planification de prises les plus utilisés sont rappelés. Tout d'abord, les différentes architectures

de préhenseur utilisées dans l'industrie ainsi que leurs principes physiques et leurs limites sont brièvement décrits. La capacité de préhension de la main humaine est abordée, ainsi que les architectures de préhenseur de la littérature qui tentent d'imiter sa versatilité, en particulier les architectures sous-actionnées. Ensuite, les principes et les limites des algorithmes de planification de prises les plus utilisés sont présentés. Enfin, les défis représentés par la conception de planificateurs de prises versatiles sont également introduits pour aider à positionner nos propres contributions dans ce domaine.

- **chapitre 2** : Le formalisme nécessaire à la description d'une prise est expliqué, notamment les matrices Jacobienne et *Grasp Map*, en mettant l'accent sur le cas du sous-actionnement. Diverses propriétés souhaitables de la prise sont présentées, et l'effet du sous-actionnement sur ces propriétés est mis en évidence. Il apparaît notamment que le sous-actionnement empêche de contrôler pleinement les torseurs d'efforts et les torseurs cinématiques aux points de contacts, et rend plus complexe l'évaluation du critère de *force-closure*. Certaines métriques de qualité de la prise sont décrites, ainsi que leurs avantages et leurs limites. La métrique de qualité choisie pour ce travail, la valeur singulière minimale de la *Grasp Map*, est également présentée et justifiée.
- **chapitre 3** : Le raisonnement à l'origine de notre contribution principale est présentée : les hypothèses envisagées, ainsi que les entrées et sorties attendues de l'algorithme. Ensuite, les principes et techniques concernant les représentations d'espaces latents et les modèles génératifs sont rappelés, avec un focus sur les Auto-Encodeur Variationnels (Variationnal Auto-Encoders, VAE), l'outil principal utilisé dans ce travail. Enfin, la méthode d'exploration de l'espace des prises elle-même est décrite. Cette méthode est objet-dépendante, et utilise des prises primitives spécifiées par démonstration pour guider l'exploration de l'espace des prises.
- **chapitre 4** : La méthodologie précédente a été spécifiquement ajustée et implémentée dans le cas d'un bras manipulateur équipé d'un préhenseur sous-actionné. Les valeurs des hyperparamètres du VAE, en particulier la dimension de l'espace latent, sont sélectionnés grâce à plusieurs essais d'apprentissage et expériences simulées. Enfin, les résultats d'expériences de planification de prises menées en simulation ainsi que sur le robot réel sont présentés.
- **chapitre 5** : Une variante de la méthode proposée, utilisant des informations issues de la géométrie de l'objet, est proposée. En effet, le principal inconvénient de la solution initialement décrite est qu'un VAE distinct doit être entraîné pour chaque objet. L'utilisation des informations de la géométrie de l'objet devrait permettre d'utiliser un seul VAE pour plusieurs objets appris, et pourrait également ouvrir des perspectives pour une extension aux objets sans prise primitive. Dans un premier temps, le formalisme concernant la représentation de la géométrie des objets est décrit. Ensuite, l'approche choisie pour intégrer ces informations à la méthode décrite précédemment est expliquée, et les différentes méthodes explorées

pour extraire efficacement les informations de la géométrie de l'objet sont présentées. Enfin, la méthode d'extraction d'informations choisie est décrite et évaluée, ainsi que la solution complète de génération de prises qui en résulte.

Cette thèse a donné lieu à deux publications dans des conférences internationales, à une soumission à une revue internationale (en cours d'examen) et à une demande de brevet :

C. ROLINAT, M. GROSSARD, S. ALOUI et C. GODIN. "Human Initiated Grasp Space Exploration Algorithm for an Underactuated Robot Gripper Using Variational Autoencoder". In : *2021 IEEE International Conference on Robotics and Automation (ICRA)*. 2021, p. 2598-2604. DOI : 10.1109/ICRA48506.2021.9561765

C. ROLINAT, M. GROSSARD, S. ALOUI et C. GODIN. "Learning to Model the Grasp Space of an Underactuated Robot Gripper Using Variational Autoencoder". In : *IFAC-PapersOnLine*. 19th IFAC Symposium on System Identification SYSID 2021 54.7 (2021), p. 523-528. ISSN : 2405-8963. DOI : 10.1016/j.ifacol.2021.08.413

C. ROLINAT, M. GROSSARD, S. ALOUI et C. GODIN. "Grasp Space Exploration Method for an Underactuated Gripper Using Human Initiated Primitive Grasps". Submitted to *International Journal of Intelligent Robotics and Applications*, under review

C. ROLINAT, M. GROSSARD, S. ALOUI et C. GODIN. "Méthode de génération de données pour la commande d'un préhenseur d'un système robotisé". Brev. FR2011310. CEA. 4 nov. 2020

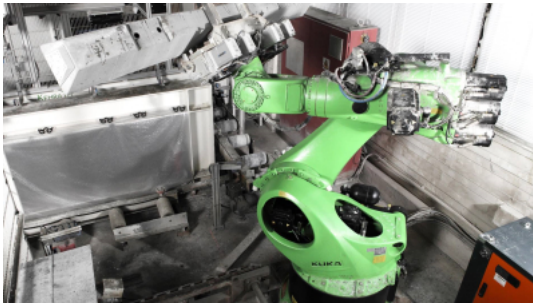
Introduction

Context

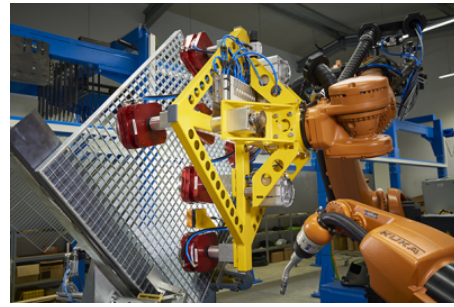
Grasping and manipulation are essential components in robotics: the gripper is at the interface between the robot and the object of interest. Manipulation requires grasping capability, and refers to the reorientation and repositioning of a grasped object relative to a given reference frame. **Grasping and manipulation abilities are required in most industrial manufacturing processes** as they often involve pick-and-place, assembly or bin picking tasks at one stage or another. **Robots are primarily introduced in the manufacturing process to avoid occupational incident, monotonous work and psychological strain for operators.** More specifically, some possible applications of robotic grasping in manufacturing industry are for example: tending of CNC (Computer Numerical Control) machine tools for workerless shift, palletizing, and object lifting and handling for ergonomic, cleanliness (for food, pharmaceutical or semiconductor industries), or safety reasons [1].

Historically, in manufacturing industry, **robots have been developed to replace humans for simple, repetitive and difficult tasks.** Robots have been introduced firstly to these types of tasks as they generally are the easiest ones to automatize, represent the greatest risk for the safety and health of the operator, and benefit the most from the cadence increase allowed by the substitution of the human by the robotic device. In this context, **robots have been specialized depending on the task.** This specialization strategy has been used in multiple components and aspects of the robotic setup. For example, regarding the robot mechanical architecture, delta robots, also known as parallel link robots, are designed for tasks requiring high precision and high execution speed, while serial manipulators are more fitted to tasks that requires a large workspace. Another example relates to the instrumentation of the robotic cell: different sensors can be required depending on the task. **The same specialization strategy is also visible for robots used in grasping and manipulation tasks, in the choose and design of gripper mechanical architecture and associated software.**

Indeed, robot grippers and grasp planning algorithms are designed specifically for the target task, given the physical properties of the object and uncertainties concerning its location and geometry [1, 2]. Some examples are given in Figure 1. In these examples, **the robots are placed in a highly controlled environment, and the correct execution of the grasping task relies on strong hypotheses about the object**



(a) Grasping of a railway sleeper with a multi-points gripper designed to fit its shape [12]. Picture: copyright ©2021, KUKA



(b) Handling of metal railings with a magnetic gripper [13]. Picture: copyright ©2021, Goudsmit



(c) Robot used in a palletizing system equipped with a suction gripper with four suckers [14]. Picture: copyright ©2021, KUKA



(d) Gripper constituted of two jaws used to pick battery cells [15]. Picture: copyright ©2021, ABB

Figure 1 – Examples of task specific industrial grippers, that use different physical principles to produce the grasp.

to be grasped. A change in the object physical properties or in the spatial arrangement of the robotic cell may need changes of the gripper architecture or its grasp planner by a human expert. This is an important drawback that prevents robots from being more widely used in industrial tasks requiring grasping or manipulation capabilities.

Recently, the framework of Industry 5.0 tends to question this specialization strategy. Indeed, the goal of this concept is to promote a more sustainable, human-centric (toward both end-users and workers), and resilient industry [3, 4]. In particular, it puts forward the concept of mass-personalisation, that aims at tailoring each product to each customer needs. It also puts the human operator at the center of the value chain, using among others, cobots (collaborative robots) designed to work seamlessly in cooperation with a human [5]. An example of such cooperation is shown on Figure 2. To achieve industry 5.0 goals, a more versatile approach of robotics is needed. **This need for versatility (Definition 1) is also present in the particular case of robotic grippers and grasp planning algorithms:**

- On one hand, current grippers struggle to manage high variability in the object physics properties (such as its shape) or expected pose relative to a given known



Figure 2 – A cobot collaborating with an operator for a small parts assembly task [16].
Picture: copyright ©2021, ABB

frame. When this kind of versatility is required, the human grasping ability and dexterity (Definition 2) remain unrivalled. This is often the case when a high level of personalisation is needed in the product. However, depending on the task and on the nature of the object, this can bring some corporal risks for the human operator, such as a higher chance to develop musculoskeletal disorders, or an increased rate of occupational accidents. To mitigate this issue, cobots can be used. **To assist more efficiently and seamlessly the human operator during a grasping task which requires fine object manipulation, they need versatile grippers and grasp planning strategies that have manipulation abilities close to the human hand.** A dexterous robotic gripper can also be helpful to perform object manipulation, even if it is not mandatory, as the robotic arm can be used to perform a manipulation task.

- On the other hand, for each task, the corresponding gripper needs to be chosen among existing grippers, or designed from scratch, and likewise for its associated grasp planning algorithm. This can represent a significant cost when developing a production line for a new product. **Having versatile grippers could help reducing the integration time and cost when creating new manufacturing lines, or when re-purposing existing ones** [6]. Like Hardware, a major need is also to bring more versatility into grasp planning strategy. These improvements allow to adapt more easily to supply chain disruptions or changes in customers' needs.

Having a robot with versatile grasping ability requires the gripper mechanical architecture to be carefully designed. **It has to offer a sufficient number of degrees of freedom to produce a wide variety of grasps for various object geometries.** In this context, a pluri-digital architecture is often chosen. However, **the architecture needs to be kept as simple as possible for cost and controller complexity**

constraints. One possible way to achieve this is to introduce underactuation (Definition 3) in some parts of the gripper mechanical architecture. **The grasping ability of the robot is also determined by its grasp planning algorithm, which itself highly depends on the chosen gripper architecture.** Indeed, the grasp planning algorithm goal is to choose from different possible grasps, and decides how a given object should be grasped in a given situation. The available grasps depend both on the object and on the gripper architecture, and on environment or task constraints. **To choose a grasp appropriate to the situation among all possible grasps, the grasp space (Definition 4) needs to be explored.**



Definition 1 (Versatility). In the framework of robotic grasping, defines the gripper ability to perform a wide variety of different tasks, and to adapt to various object geometries. This definition is derived from the one given in Gazeau [2].

Definition 2 (Dexterity). Ability to move a grasped object relative to the gripper or hand frame along a given trajectory, while maintaining the object stability. This definition comes from Gazeau [2].

Definition 3 (Underactuation). Property of a system to have an input vector of smaller dimension than the output vector. In a robotic context, it means having fewer actuators than degrees of freedom. This is the definition given by Birglen et al. [7].

Definition 4 (Grasp space). In this work, this concept refers to the set of all possible grasps that a given gripper can produce on a given object. This space is a subset of the gripper configuration set.

Outlines & Contributions

This thesis brings several published contributions that leads to the proposition of a general software framework for **exploring the grasp space of robotic gripper, that is particularly efficient in the case of pluridigital and underactuated architectures.** It uses Variational Auto-Encoders to generate grasps, together with a prediction of their quality, taking inspiration from human-provided primitive grasps.

The structure of this manuscript, shown in Figure 3, is divided as follows:

- **chapter 1:** Commonly used state-of-the-art gripper architectures and grasp planning algorithms are recalled. First, the different gripper architectures used in the industry along with their physical principles and limitations are shortly described.

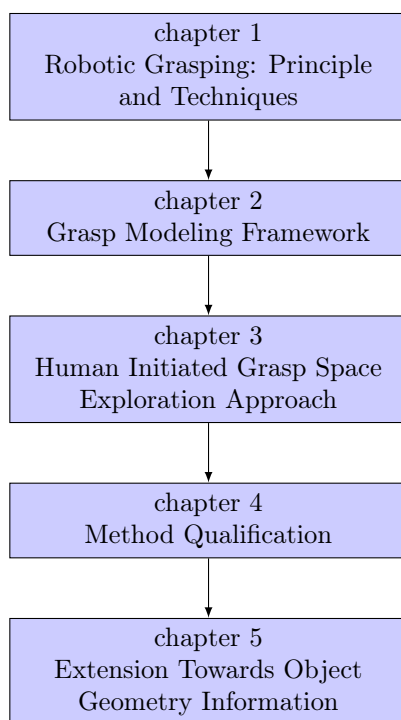


Figure 3 – Organisation chart of the manuscript.

The grasping ability of the human hand is discussed, together with state-of-the-art gripper architectures that attempt to imitate its versatility. Then, the principles and limitations of most used grasp planning algorithms are presented. Finally, the challenges represented by the design of versatile grasp planners are also introduced to help positioning our own contributions in the field.

- **chapter 2:** The formalism needed for the description of a grasp is explained, with a focus on the underactuated case. Various grasp desirable properties are presented, and the effect of the underactuation on these properties is highlighted. Some grasp quality metrics are described, together with their benefits and limitations. The quality metric chosen for this work is also presented and justified.
- **chapter 3:** The rationale behind our main contribution is presented: the considered hypotheses, and the expected input and output of the presented algorithm. Then, principles and techniques regarding latent space representations and generative models are recalled, with a focus on VAE, the main tool used in this work. Finally, the method itself is described.
- **chapter 4:** Previous methodology has been specifically tuned and implemented in the case of an underactuated picking station. Specifications for the VAE hyperparameters, in particular the latent space dimension, are drawn thanks to several

learning trials and simulated experiments. Finally, the results of grasp planning experiments conducted in simulation as well as on the real robot are given.

- **chapter 5:** A variant of the proposed method, using information from the object geometry, is presented. Indeed, the main drawback of the proposed framework is that a distinct VAE needs to be trained for each object. The use of object geometry information should allow to use a single VAE for multiple learned objects, and might also open up perspectives for an extension to object without primitive grasps. First, the formalism regarding object geometry representation is described. Then, the chosen approach to integrate this information to the method previously described is explained, and the various explored methods to efficiently extract information from the object geometry are presented. Finally, the chosen information extraction method is described and evaluated, along with the resulting complete grasp generation workflow.

This thesis resulted in two publications in international conferences, one submission to an international journal (still being reviewed), and one patent application:

C. Rolinat, M. Grossard, S. Aloui, and C. Godin. “Human Initiated Grasp Space Exploration Algorithm for an Underactuated Robot Gripper Using Variational Autoencoder”. In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. 2021, pp. 2598–2604. DOI: 10.1109/ICRA48506.2021.9561765

C. Rolinat, M. Grossard, S. Aloui, and C. Godin. “Learning to Model the Grasp Space of an Underactuated Robot Gripper Using Variational Autoencoder”. In: *IFAC-PapersOnLine*. 19th IFAC Symposium on System Identification SYSID 2021 54.7 (2021), pp. 523–528. ISSN: 2405-8963. DOI: 10.1016/j.ifacol.2021.08.413

C. Rolinat, M. Grossard, S. Aloui, and C. Godin. “Grasp Space Exploration Method for an Underactuated Gripper Using Human Initiated Primitive Grasps”. Submitted to *International Journal of Intelligent Robotics and Applications*, under review

C. Rolinat, M. Grossard, S. Aloui, and C. Godin. “Méthode de génération de données pour la commande d’un préhenseur d’un système robotisé”. Pat. FR2011310. CEA. Nov. 4, 2020

Chapter 1

Robotic Grasping: Principle and Techniques

In this chapter, grasp planning techniques are discussed after an introductory part on the gripper architecture, since existing planning approaches are strongly hardware-dependent. First, existing industrial architectures for grasping are described, and the need for versatility is discussed. Then, the grasp planning topic is introduced, and various techniques are presented. Finally, a focus is made on one of the key component for versatile grasp planning: the grasp space exploration. The issue that it represents for underactuated gripper is also discussed.

Contents

1.1	Grasping in Industry: Overview	20
1.1.1	Generalities	20
1.1.2	Gripper Classification	24
1.1.3	Limitations	29
1.2	Toward More Versatile Grippers	29
1.2.1	The Human Hand Example	29
1.2.2	Dexterous and Versatile Multifingered Grippers	32
1.2.3	Underactuated Grippers	36
1.3	Grasp Planning Principles	39
1.3.1	Problem statement	39
1.3.2	Overview of Grasp Planning Approaches	40
1.3.3	Limitations	44
1.4	Grasp Space Exploration Issue	45
1.4.1	Definition	45
1.4.2	Grasp Space Exploration with Underactuated Grippers	49
1.5	Conclusion	51

1.1 Grasping in Industry: Overview

1.1.1 Generalities

Grasping abilities can be useful for a wide variety of applications, which lead to the development of numerous grippers and associated robotic environment.

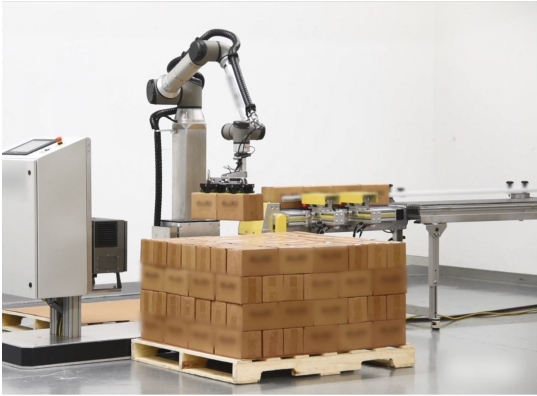
Applications

In a manufacturing process, grasping can be required at multiple stages. Indeed, various objects need to be grasped and handled to go from raw materials to end products.

- First, grasping is needed right before and right after the production line itself, that is for intralogistic operations. More specifically, grasping is required for bin picking, bin packing, palletizing and order picking.
 - Palletizing refers to stacking (or unstacking) boxes on a pallet (Figure 1.1a). The stacking process occurs at the end of the production line, before shipment or storage in a pallet racking. For the unstacking case, it occurs mostly when receiving raw materials, and may be followed by a bin picking stage.
 - Bin packing is the process of putting items in a box, before palletizing it for example (Figure 1.1b).
 - Conversely, bin picking refers to grasping objects having random and a priori unknown poses out of a bin (Figure 1.1c). This is typically the case for raw materials or parts: they may be received loose in a storage box. In order to supply properly the production line, they need to be picked out of the box one by one.
 - Order picking is a process that has known an important development with the growth of the e-shopping sector. The goal is to fulfill a customer's order, by picking individual items from their storage locations, gather them, and ship them to the customer (Figure 1.1d).
- Then, grasping is also needed during the production process itself. Indeed, a production line can contain a significant number of pick-and-place tasks:
 - handling objects from one pose to an other,
 - tending of a CNC (Computer Numerical Control) machine tool or special machine [17],
 - assembly tasks.

Most of the times, the different required parts are moved in the production line by conveyor belts or similar systems. However, some re-positioning or re-orientation can be difficult or impossible to achieve with a conveyor, and the parts need to be handled. Regarding CNC machine tools, even if they do not require any live

manipulation to control them, the part still needs to be set in place in the machine, and get back once the process is finished. Concerning assembly tasks, it requires advanced grasping abilities so that the part can be handled properly and fit correctly in its place.



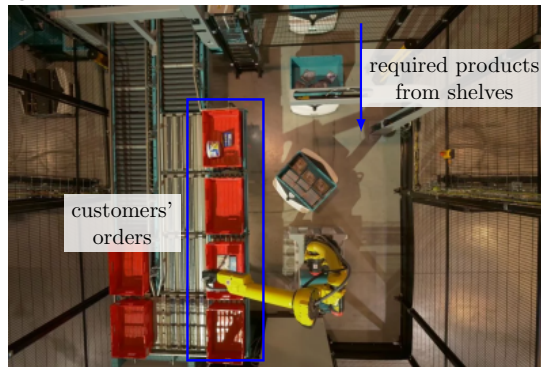
(a) Example of a robotized palletizing station. Picture: CC BY-SA license, adapted from [18]



(b) Bin packing system. Picture: copyright ©2022, RECOMI Industrie [19]



(c) Example of a bin picking task. Picture: copyright ©2022, romi-is.com [20]



(d) Robotized order picking station commercialized by EXOTEC [21]. Picture: adapted from [22], copyright ©2020, L.A.C. Conveyors & Automation

Figure 1.1 – Various intralogistic operations that require grasping abilities.

When these tasks are performed by a human operator, they can represent a significant risk for his health. Indeed, their repetitiveness can be a ground that favors the development of musculoskeletal disorder or psychological strain [23, 24]. Both issues can cause occupational accident, or non-quality in the production due to human error. Regarding issues linked to mental fatigue, techniques are developed to monitor an operator fatigue state, such as in Charbonnier et al. [25], to prevent incident or accident linked to a lack of vigilance. However, the best way to limit mental fatigue is still to avoid as much as possible any repetitive and tedious tasks. Thus, robotizing these grasping tasks can help to solve these issues.

Some manufacturing sectors can take even more advantages from the automation of grasping tasks. In particular, industries that require a high level of cleanliness, for example food [26, 27], pharmaceutical, or semiconductor [28], and industries that involve the handling of toxic and dangerous materials, like chemical or nuclear industries [29] (see Figure 1.2). Indeed, cleaning, clothing and more generally safety procedures are constraining and prone to human errors, that can provoke quality or safety issues which can put worker's or customer's health at stake. Robotic grasping can help to isolate the product from the human, or at least take them away from each other.



Figure 1.2 – ROMANS European project [30]. It aims at developing robotic solution for nuclear waste sorting and segregation. One of the proposed solution is a bi-manual master-slave tele-operated system.

Robotic Architecture for Grasping

In a manufacturing context, robots are implemented through robotic cells. In addition to robotic manipulators, several other automated components can be installed to fulfill the task requirement, such as conveyor belts, CNC machine tools, or any other automated special machines. A wide variety of sensors can also be present in the robotic cell. Examples of robotic cells are shown on Figure 1.

Sensors can be divided in two main categories: proprioceptive and exteroceptive sensors.

- Proprioceptive sensors can measure information about the internal state of the system, for example, the position or velocity of a given actuator or joint. Torque sensors can also be used on the joints, allowing safe interaction with humans for cobots.
- Exteroceptive sensors can measure the state of the environment. One of the simplest example are optoelectronic sensors. They can detect the presence or absence of an object in front of them, and can be used to trigger simple processes when the environment is well known. 2D and 3D cameras can give way richer information: it is often used to determine the object pose when it is not fully known, thanks

to object detection or segmentation algorithms, coupled with pose estimation algorithms [31, 32] (see Figures 1.3a and 1.3b). The camera can be installed on the robotic arm, in a *eye in hand* configuration, or fixed in the robotic cell, in a *eye to hand* configuration. An other example of exteroceptive sensor are force-torque sensors. Such sensors can be useful for example in assembly tasks, when the pieces need to be put together with a given force or torque. Various other sensors can be installed in the robotic cell, for quality control for example.



(a) Integrated vision system proposed by an industrial robot manufacturer for part location, inspection and identification. The system includes image processing algorithms that are executed directly by the robot controller. Picture: copyright ©2021, ABB



(b) Multispectral 3D camera for object pose estimation and quality control. It is shipped with its own controller and image processing algorithms. Picture: copyright ©2021, Tridimeo

Figure 1.3 – Examples of vision sensors that can be used for grasping applications.

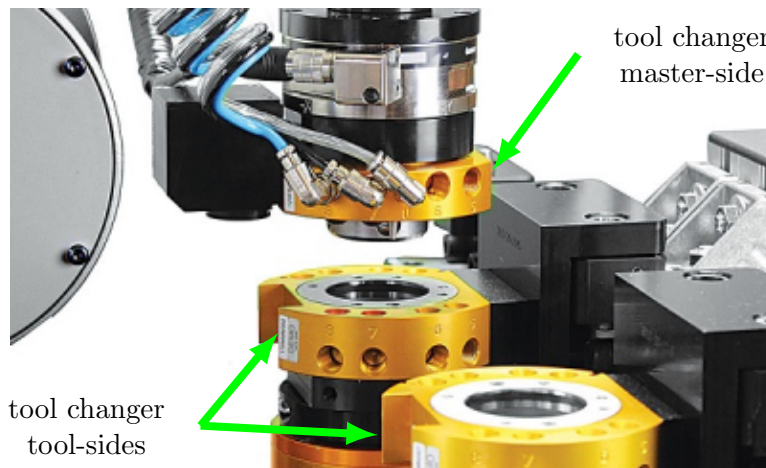


Figure 1.4 – Example of a tool changer. Picture: copyright ©2021, DESTACO

Gripper mechanical architectures are chosen depending on the task, and are often

highly specialized, as it is shown in subsection 1.1.2. When several objects with different geometries and physical properties need to be handled in the same task, finding a gripper suitable for all objects may be impossible. In that case, two possibilities remain: using several robots, each one with a different gripper, or using one robot with a tool changer. The first possibility is by far the most complex and expansive, but may allow the highest production rate. The second option is cheaper, but also slower, as each tool change can take several seconds. Their main working principle involves the following components (see Figure 1.4): a master-side, fixed on the robot flange, and several tool-sides, each one fixed to a different tool, the master-side being able to connect with any tool-side.

1.1.2 Gripper Classification

To grasp an object, a sufficient effort needs to be applied on it, first to overcome gravity, and also any dynamic wrenches that can be produced during the execution of the task. Several physical principles are used to achieve this, and the various types of existing gripper architectures can be divided based on their principle of operation [33]:

- **astriuctive group:** a non-contact force is applied in one direction to grasp the object;
- **impactive group:** contact forces in at least two directions are used to perform the grasp;
- **ingressive group:** the grasp is achieved by permeation of the object surface;
- **contigutive group:** the grasp is performed by applying a contact force in one single direction.

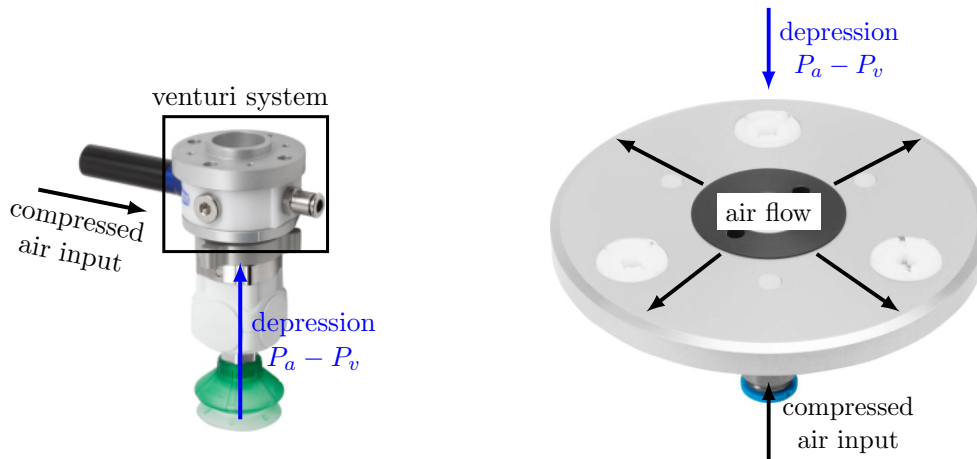
Astriuctive Grippers

This type of grippers is widely used for pick-and-place, palletizing, bin picking or bin packing tasks. There are three main categories: suction, magnetic and electrostatic grippers.

Suction Grippers The grasping effort is produced by creating a depression between a suction element on the gripper and the object. Those grippers are constituted of one or several suckers depending of the object characteristics (an example on a palletizing task is shown on Figure 1c).

The depression can be produced by two mechanisms [2]:

- **Venturi effect,** where compressed air go through a conical nozzle which increases the air flow velocity, and thus create a depression. An example of sucker using Venturi effect is displayed in Figure 1.5a.
- **Bernoulli effect,** where a depression is created thanks to an increased air flow velocity in the space separating the sucker from the object, caused by the injection of compressed air along the sucker. Such sucker is shown in Figure 1.5b. With this



(a) Suction gripper using Venturi effect. Picture: copyright ©2021, Schmalz

(b) Suction gripper using Bernoulli effect. Picture: copyright ©2021, Festo

Figure 1.5 – Examples of suction grippers. For a suction element, the grasping force is given by $F = (P_a - P_v)A$, with A the contact surface, and $P_a - P_v$ the depression. P_a is the atmospheric pressure, and P_v the working pressure (a negative relative pressure).

technique, there is very little contact surface between the object and the sucker, which is suitable for fragile objects.

In both cases, the created working pressure P_v , that produces the depression, depends on the compressed air input pressure: the higher the input pressure is, the lower the working pressure. Suction grippers allow very fast grasping, as the time needed for object adhesion is under 10 ms. Moreover, the produced suction force allows to lift relatively heavy object, with around 50 kN/m^2 [34].

Magnetic Grippers This type of grippers can only work on ferromagnetic materials, such as iron, nickel, cobalt and their alloys, and to a smaller extent on paramagnetic materials, such as aluminium and lithium. They use a magnetic field to produce an electromagnetic force that lifts the object. They are often used to handle large or heavy metal pieces, or metal pieces with a lot of holes. Indeed, contrary to suction grippers, the attraction force of magnetic grippers does not depend on object surface, but on its magnetization property, which is a volumetric property. Moreover, the produced force can be very strong, around 100 kN/m^2 [34]. An example of such gripper is displayed in Figure 1b. They are divided in two main categories [2]:

- Permanent magnet grippers, that are themselves divided in two sub-categories, depending on the type of power needed for activation:
 - electric commutation permanent magnet grippers use two identical permanent magnets that rotate relative to each other to commute. A scheme of this mechanism is shown in Figure 1.6a.

- Pneumatic commutation permanent magnet grippers use compressed air to move a permanent magnet inside a cavity. A scheme of this type of commutation is displayed in Figure 1.6b.

These types of systems allow unpowered attraction, which is an advantage in terms of safety.

- electromagnet grippers, that use an electromagnet to generate the magnetic field needed to create electromagnetic force. This type of gripper allows only powered attraction: this can represent a risk in case of power failure, and an additional locking system is often needed.

Electrostatic Grippers This type of grippers use electrodes charged at high voltage to generate an intense electric field. The surface of surrounding conductive materials becomes charged, while the surface of surrounding insulated materials becomes polarized. In both cases, an attractive force appears between the gripper electrodes and the object surface [34]. This force is significantly smaller than for suction or magnetic gripper, at around 100 N/m^2 . For that reason, it is mainly used to handle very light objects, such as paper sheets or textile fabrics, or also for micro or nano materials manipulation.

These three types of grippers work in an all or none manner, and do not allow to detect the correct grasping of the object without additional sensors [2], even if smart Venturi systems developed recently allow such detection by integrating the required sensors and electronics.

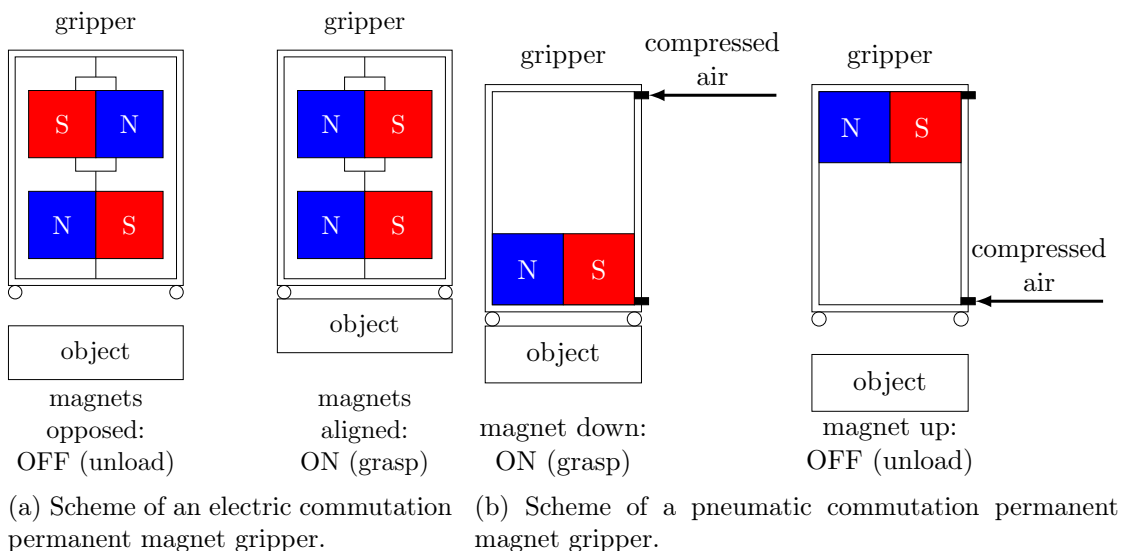


Figure 1.6 – Schemes of permanent magnet gripper principles.

Impactive Grippers

Those grippers rely on contact mechanics, and are well suited for custom grasping and handling tasks of objects with complex shapes. They can be divided into two main categories:

- Parallel grippers that are constituted of two symmetric fingers, and exploits the friction existing between the object and the fingers (see on Figure 1d for example). The grasp is produced by closing the fingers on the object and exerting a sufficient tightening force on it to overcome weight and inertial forces. These grippers work best on objects having polyhedral or prismatic geometries, with some parallel faces where the gripper fingers can fit.
- Multipoint grippers, that have more than two fingers, or are constituted of one or several jaws designed to fit the shape of the object (an example is displayed on Figure 1a). This way, the grasp depends less on friction forces, as the object can be geometrically locked in the gripper. These grippers rely less on the presence of parallel faces on the object, and can be designed to fit an arbitrary geometries.

A more in depth description of the physics of grasps using contact mechanics is given in chapter 2. These grippers can be powered by pneumatic, electric, or hydraulic energy:

- Hydraulic energy allows very high tightening efforts, but is used in grasping tasks almost exclusively in the construction industry or other specific industrial sectors requiring high efforts, and not in manufacturing robotics. This is due to the complexity of such systems, and the limited efforts typically required in manufacturing tasks.
- Grippers using pneumatic energy use linear or rotary actuator supplied with compressed air. The tightening effort is controlled by adjusting the compressed air input pressure. Thus, this effort is adjusted when setting up the robotic cell, and is usually not modified during the task cycle. Grippers powered by compressed air operate in an all or none manner, and cannot detect correct grasping without supplementary instrumentation. As suction grippers, they allow fast grasping, thanks to very short closing and opening times of around 10 ms [2].
- Electric grippers can be divided in two categories: classic mechanically actuated grippers, or piezoelectric grippers.
 - Mechanically actuated grippers allow to control precisely the velocity and position of the fingers during closing or opening, thanks to one or several actuators. An effort control is also possible if the transmission system is transparent enough. The object grasping can be detected without additional sensors as a disturbance on the position of the jaws. However, compared to pneumatic grippers, electric ones are generally bulkier, heavier, and slower (with an opening/closing time between 50 and 500 ms [2]).

- Piezoelectric grippers use piezoelectric actuators, that exploit the property of some materials to deform themselves when surrounded by an electric field. These grippers are cheaper, simpler and lighter than classic ones, but generally cannot produce as high grasping forces as them. They are well suited for handling micro and nano parts. However, the control of such grippers is more complex, as piezoelectric actuators can exhibit unstable behaviors [35].

Ingressive Grippers

The main application of ingressive grippers is for grasping fabrics or in the handling of carbon fibers [35]. These grippers grasp fabrics by penetrating them with a spiked wheel, or with straight or bent needles. Indeed, most textile materials are too soft to be efficiently grasped with impactive grippers, and often cannot be grasped with astrictive gripper either, while they can be penetrated and moved without damaging the woven structure.

Contigutive Grippers

These grippers are based on chemical or thermal adhesion, and need a contact between the object and the gripper to be established beforehand, contrary to astrictive grippers. When based on chemical adhesion, the gripper uses glue or sticky adhesive to grasp the object. For thermal adhesion, a surface can be melted to adhere to the object, or conversely, a thin water layer between the object and the gripper can be frozen. Contigutive grippers are mainly used for fabrics handling, or for micro and nano parts manipulation [35].

The main use cases of these different gripper categories (astrictive, impactive, ingressive, contigutive) are summarized in Table 1.1.

Types of Objects	Type of Grippers			
	Impactive	Ingressive	Astrictive	Contigutive
Solid Flat Objects	●●		●●	●
Solid Curved Objects	●●		●●	
Solid Irregular Shapes	●●			
Flexible Sheets	●	●●	●	●
Rigid Sheets	●		●●	●
Fragile Objects	●		●●	
Micro and Nano Objects	●●	●	●	●

Table 1.1 – Comparison of the ability of grippers to pick up different types of objects (●●: commonly used, ●: sometime used) [35].

1.1.3 Limitations

As shown in subsection 1.1.1, robotic grasping has multiple applications in a wide range of industries, and the implementation of a robotic cell involves various components.

The various gripper architectures presented in subsection 1.1.2 shows a high degree of specialization to the targeted grasping task, as summarized in Table 1.1. The more versatile (Definition 1) are impactive and astrictive gripper, but they both have their limits.

A typical manufacturing process may require several different grippers, each specialized in a given task. This is a major issue for the resilience of the whole system. For example, numerous different spare parts need to be stored, and maintenance operators need to have a various set of skills and knowledge about the different grippers involved, in order to be robust to gripper failures [6]. Moreover, the high degrees of specialization of each gripper limits and complicates any evolution or improvement of the production process, or of the produced parts. Thus, versatile grippers can lead to a more resilient manufacturing industry.

The high specialization of existing grippers also prevents them from being used in a small batches or personalized manufacturing context, which will become more and more widespread with industry 5.0 [4]. Indeed, the object may vary too much between batches to find a gripper suitable for any possible variation. In such case, a human operator would be needed, as the human hand is the universal gripper par excellence. Thus, there is a need for grippers with increased versatility to automatize those tasks.

1.2 Toward More Versatile Grippers

1.2.1 The Human Hand Example



Recall the definitions of dexterity and versatility:

Definition 1 (Versatility). In the framework of robotic grasping, defines the gripper ability to perform a wide variety of different tasks, and to adapt to various object geometries. This definition is derived from the one given in Gazeau [2].

Definition 2 (Dexterity). Ability to move a grasped object relative to the gripper or hand frame along a given trajectory, while maintaining the object stability. This definition comes from Gazeau [2].

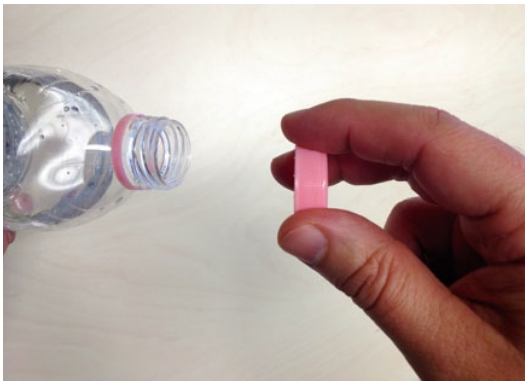
The main issue of current grippers is their lack of dexterity, and the limited number of possible grasps they offer, that is their lack of versatility.

The most versatile and dexterous multifingered gripper to date is the human hand. It is very versatile, as we are not only able to grasp a vast variety of objects, but also to perform various grasps for each of them. It is also dexterous, as for almost any object

of reasonable size, we are able to reorient it inside our hand very precisely and robustly. The human hand is also the most complex gripper to date, as it has between 21 and 25 degrees of freedom, distributed on 18 joints, controlled by 39 muscles and supported by 27 bones [38]. Human produced grasps have been classified in the literature in several grasp taxonomies. They show the ability of the human hand to produce a wide variety of different grasps [2]. An example of such taxonomy is displayed in Figure 1.8. Two main grasp categories described in Napier et al. [36] are common to most existing ones:

- power grasps, that have large contact surfaces between the hand and the object. They do not allow in hand manipulation, but are very stable and versatile (left side of Figure 1.8). In this type of grasp, the whole fingers and the palm are used to constrain the object as much as possible. An example of such grasp is shown in Figure 1.7b.
- precision grasps, that involve only the fingertips. This kind of grasps allows dexterous manipulation of the grasped object (right side of Figure 1.8). In this type of grasp, two or more fingers are placed on the object, and apply the required effort to maintain it in the hand. Typically, the thumb is placed in opposition with one or more fingers. A hand performing this type of grasp is displayed in Figure 1.7a.

The human-inspired taxonomy presented in Figure 1.8 is driven by the idea that the choice of a grasp depends both on the object geometry and on the task requirements. This specific taxonomy is not exhaustive, as it is focused on grasps found in a manufacturing context, and does not include some everyday grasps, such as writing with a pencil, or holding a key for example. Other taxonomies exist and does not focus on grasp used in a manufacturing context, for example the ones suggested by Bullock and Dollar [40] or Feix et al. [41].



(a) Hand performing a precision grasp. This grasp corresponds to the number 14 of the taxonomy in Figure 1.8.



(b) Hand performing a power grasp. This grasp corresponds to the number 1 or 2 of the taxonomy in Figure 1.8.

Figure 1.7 – Example of the two main grasp categories described in Napier et al. [36]. Pictures: copyright ©2016, Springer-Verlag Berlin Heidelberg [37]

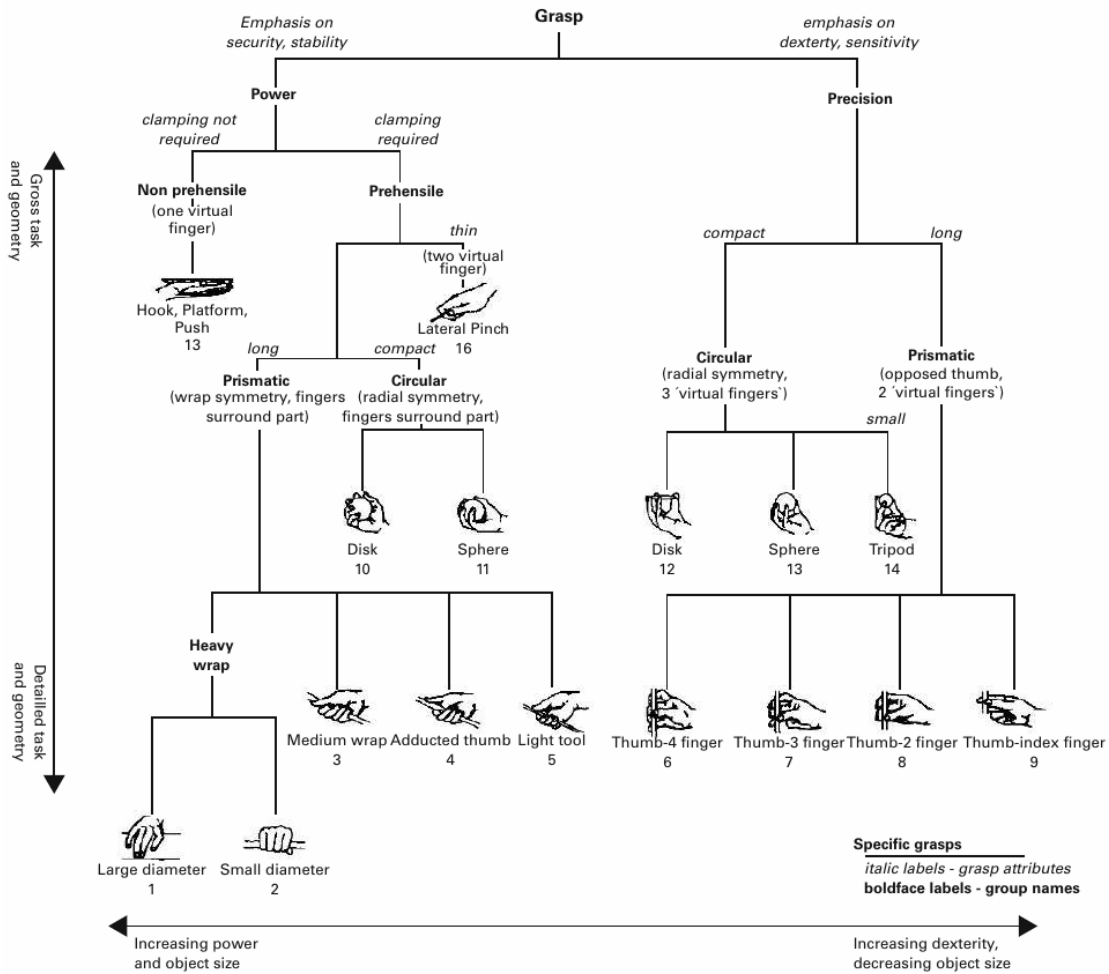


Figure 1.8 – Partial taxonomy of grasps encountered in a manufacturing context [39]. copyright ©1989, IEEE

It is worth noting that for a given object and task, the grasp chosen by a human is not static, but may evolve during the task execution. Indeed, we adapt the grasp depending on the trade-off between force/torque and dexterity required by the task. For example, when using a tool, an operator can start with grasp No. 3 when a lot of force/torque are needed, and change afterward to grasp No. 5 or even No. 6 when the required force/torque decreases, and if more dexterity is needed [39]. Moreover, humans have the possibility to use both hands, when additional force/torque is required, or to improve the dexterity of the grasp. This ability to change the chosen grasp during the task execution, or to produce bi-manual grasps, makes the human hand even more versatile.

From a control perspective, studies have been conducted to determine how our brain is able to manage such a complex system. First, a large portion of the human motor cortex, around 30% - 40%, is reserved to the control of the hand. Studies revealed that

it is controlled following principal motions, that are combined and mixed together to produce a wide variety of grasp types, some of which are shown in the taxonomy in Figure 1.8. Actually, the brain controls the hand in a configuration space much smaller than the 39 dimensional space of the hand muscles.

The concept of postural hand synergies has been introduced to describe this mechanism [38, 42]. This means that to execute a grasp, one does not need to control independently each degree of freedom of his hand. On the contrary, the numerous degrees of freedom allowed by the complex mechanical architecture of the hand are highly interdependent in the postural hand synergies space.

1.2.2 Dexterous and Versatile Multifingered Grippers

In the literature, more than a hundred multifingered grippers have been developed [2], but to our knowledge, none of them has been commercialized and used in a real manufacturing process yet. Some examples of existing multifingered grippers are displayed in Figure 1.9. Existing multifingered grippers have from three to five fingers, and up to 24 actuated degrees of freedom.

The main issue with multifingered grippers is the complexity of such grippers [43]: each controllable degree of freedom requires at least one actuator and its associated transmission system, as well as a suitable control algorithm. Nevertheless, such grippers can be better than the human hand at producing independent motion of their different joints. For example, for most humans, the second, third, fourth and fifth fingers cannot be moved completely independently, and likewise for the intermediate and distal phalanges of a given finger, whereas it may not be an issue for a fully actuated multifingered gripper [7].

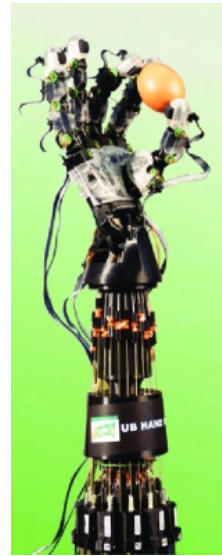
From a mechanical point of view, the bulk and cost of such system is still an issue. Indeed, a high number of degrees of freedom means a high mechanical complexity, and a high number of actuators, both factors increasing the cost of the system. Moreover, each actuator and its transmission chain needs to be integrated in the system.

To minimize the complexity of the transmission chain, it is easier to have the actuator as close as possible to the controlled joint with a transmission constituted of as few gears as possible. This is the choice made for the DLR-HIT gripper for example (Figure 1.9c). However, in case of grippers, the available space is limited inside the fingers or the palm, unless one increases the gripper size, which limits its potential applications. Therefore, integrating the actuators here allows to place only very limited number of them, which in turn allows to drive only a few degrees of freedom. Integration in a exiguous area also require to use small actuators, that can exert a limited amount of force on the joints [48].

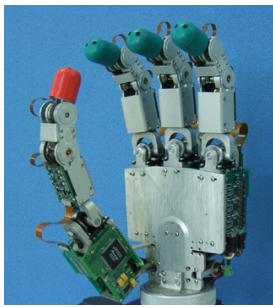
To drive more degrees of freedom, and to exert higher forces, it is necessary to place the actuators in the wrist, for example as in the RoBioSS gripper (Figure 1.9d), or even farther away in the forearm, as in the ShadowHand or Dexmart grippers (Figures 1.9a and 1.9b). This comes at the cost of a more complex drive chain, often based on tendons [2, 7, 48]. Different implementations of this actuation exist, and two main schemes can be identified:



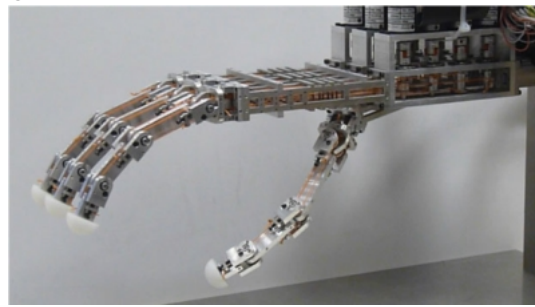
(a) ShadowHand gripper: 5 fingered gripper with 24 degrees of freedom. Picture: CC BY-SA license [44]



(b) Dexmart gripper: 5 fingered gripper with 17 degrees of freedom [45]. Picture: copyright ©2014, SAGE Publications



(c) DLR-HIT gripper: 4 fingered gripper with 13 degrees of freedom [46]. Picture: copyright ©2006, Elsevier



(d) RoBioSS gripper: 4 fingered gripper with 16 degrees of freedom [47]. Picture: copyright ©2018, Cambridge University Press

Figure 1.9 – Examples of dexterous and versatile multifingered robotic grippers in the literature.

- The simple effect scheme, where an actuator can transmit an unidirectional force to a joint. In this case, there are one actuator for each degree of freedom. Each joint has one side of its tendon linked to an actuator, and the other side attached to a passive return spring, which allows the return motion of the joint (Figure 1.10a). A drawback of this system is that a portion of the actuation power is stored in the spring, and not available to the joint. An other configuration is also possible, with two actuators for each degree of freedom. In this configuration, there is an actuator on both ends of the tendon of a joint, which enables an active return motion (Figure 1.10b). It is very expensive, and poses a compactness problem, but

allows to control joint stiffness by adjusting the tension force created by the two actuators [2, 48].

- The double effect scheme, which allows an actuator to transmit a force to the joint in both directions. In this case, there is one actuator for each degree of freedom, and both sides of the tendon of each joint are linked to the same actuator (Figure 1.10c). In that case, the preload of the system is essential. Indeed, there is always a slack side and a tight side. A too loose tendon can be an issue for the cable routing, and can lead to the appearing of hysteresis (dead zone phenomena created when the actuator changes direction) [2, 48]. Conversely, a too tightened tendon can create unnecessary mechanical stress and friction. It has been shown that a configuration with one more actuator than degrees of freedom allows to maintain tension on all tendons, similarly to the simple effect scheme, while preserving a double effect scheme [49].

Such transmission system has the advantage of being light-weighted and flexible to use compared to rigid elements, but also can bring several issues: friction, hysteresis, elastic behavior due to the use of finitely-rigid cables, or undesired kinematic coupling between joints due to the tendons routing [2, 7, 48]. Regarding internal friction and undesired joint coupling, significant improvements can be obtained by an appropriate design of the cable routing, such as in the RoBioSS gripper for example [47].

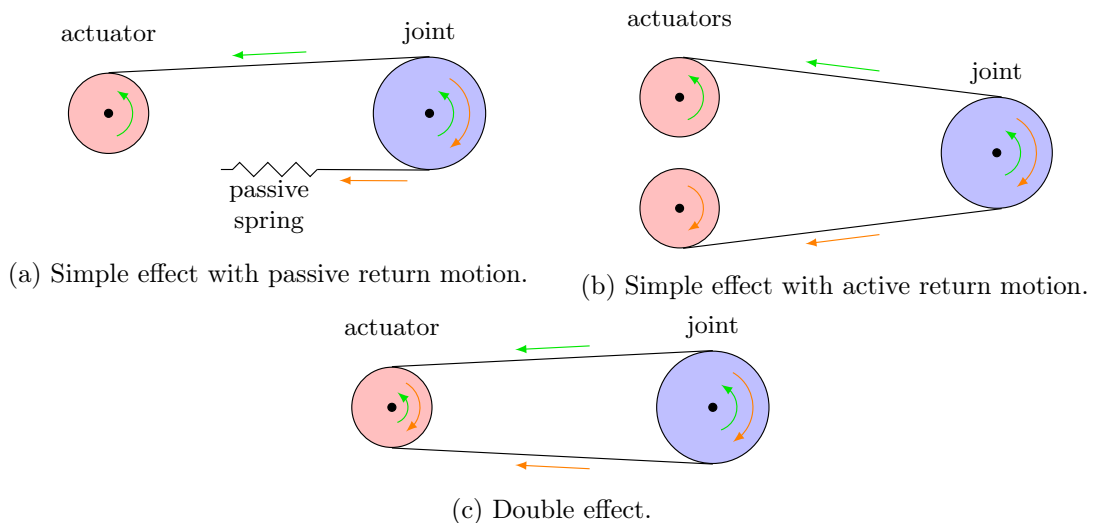


Figure 1.10 – Summary of the main tendon-based actuation schemes. The forces and torques transmitted from the actuator(s) (or spring in case of the passive return motion) to the joint in both directions are represented in green and orange respectively.

From a control point of view, the aforementioned friction and coupling issues need to be taken into account, which make the controller more complex. The high number of degrees of freedom also contributes to this increased complexity: the trajectories of the

numerous joints of the gripper need to be planned (and then controlled) simultaneously to grasp and manipulate an object. A possible way to simplify the controller is by taking inspiration from human postural hand synergies, which allow to control the hand in a smaller space than the actuation space. However, transferring such synergies on a robotic gripper is not trivial: the human hand kinematics admits interpersonal variations, and the robotic gripper kinematics can present important dissimilarities with it [38, 42, 45].

Palli et al. [45] and Monforte et al. [38] proposed a postural synergy based controller for anthropomorphic hands. The process is summarized in Figure 1.11. First, it requires to capture human hand movements when grasping various objects (step 1b), which can be done by instrumenting the hand or using image processing (or both), to retrieve the pose of the hand palm and fingertips (step 2). The full hand configuration can be retrieved using an inverse kinematics algorithm (step 3b) that use a rescaled version of the anthropomorphic gripper as the reference kinematic model. The rescaling is computed beforehand (step 3a) from measurements of the subject's hand that performed the grasps (step 1a). Then, human postural hand synergies themselves need to be computed, with dimension reduction tools, such as Principal Component Analysis (PCA), applied on the hand configurations corresponding to the registered grasps (step 4). Finally, the computed postural synergies can be used to replicate with the robotic gripper the grasps and/or the in hand manipulation performed by the human.

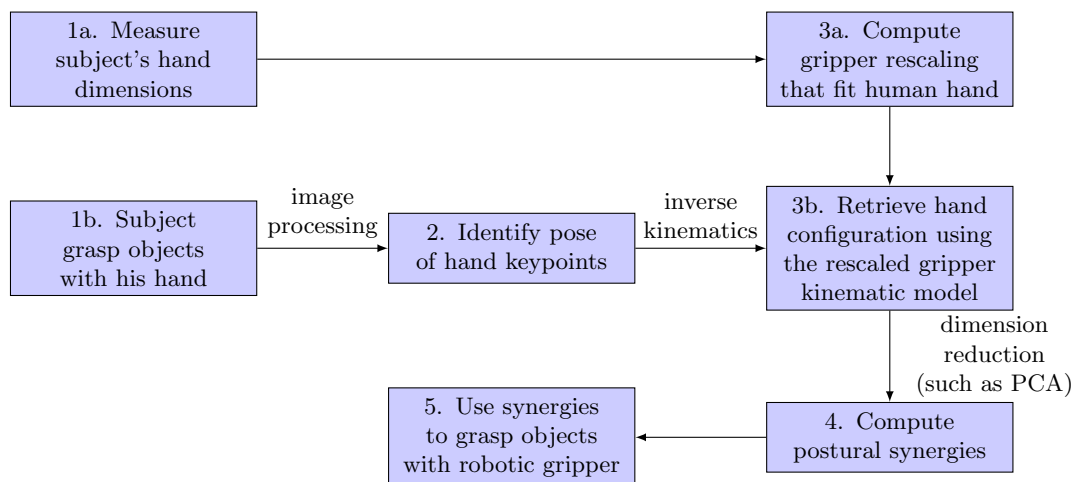


Figure 1.11 – Scheme summarizing a human-cognition inspired multifingered gripper controller proposed in [38, 45].

If the gripper is not anthropomorphic, such methods based on joint space synergies cannot be used. The postural synergies principle can still be used, but in the task space instead (that is the space of object trajectory and wrench), such as proposed by Gioioso et al. [50]. The authors identified a set of task space synergies from a paradigmatic human hand and mapped them to a robotic gripper, without any constraints on the gripper architecture. An other approach to control non-anthropomorphic gripper that does not use postural synergies is proposed by Romero [42]: gripper preshapes corresponding

to different human grasp types (for examples from the taxonomy in Figure 1.8) are handcrafted beforehand, and triggered depending on the object. However this later approach is limited to grasping task, and does not allow dexterous manipulation.

An other issue with complex multifingered grippers is that they might be overdesigned when considering independently the grasping issue. Their complexity allows them to perform a dexterous manipulation of the object, but can also bring robustness issues, from both the mechanical structure and the controller. They are still able to grasp objects efficiently, but it can be more straightforward and robust to perform a grasp using simpler grippers, such as for example the ones presented in subsection 1.1.2. This is known as the *grasping vs manipulating* dilemma: a gripper designed for grasping will be, in general, unable to manipulate an object, while a gripper conceived for manipulation might not be the optimal choice for a grasping task that does not requires fine in-hand manipulation [7].

1.2.3 Underactuated Grippers



Recall the definition of underactuation:

Definition 3 (Underactuation). Property of a system to have an input vector of smaller dimension than the output vector. In a robotic context, it means having fewer actuators than degrees of freedom. This is the definition given by Birglen et al. [7].

There exists grasping tasks that involve little or no fine manipulation abilities. For these tasks, a dexterous gripper may not be mandatory. To retain the versatile behavior inherent to multifingered grippers, while avoiding the complexity of a high degree of actuation, two techniques are predominantly used: joint coupling and underactuation. Both techniques can be found in several part of a given gripper, and come as a trade-off between dexterity and simplicity, while retaining versatility.

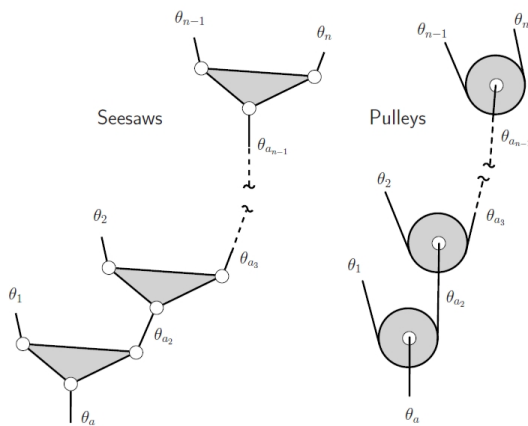
The concept of joint coupling is not limited to grippers, and refers to a mechanism where some joints are not independently controlled by a given actuator. Mainly two forms of coupling can be identified:

- a joint is driven by several actuators;
- an actuator drives several joints. It allows a mechanism to produce complex trajectories, with less degrees of freedom than joints.

Joint coupling often appears as an undesirable byproduct of a transmission system, but can also be a useful feature.

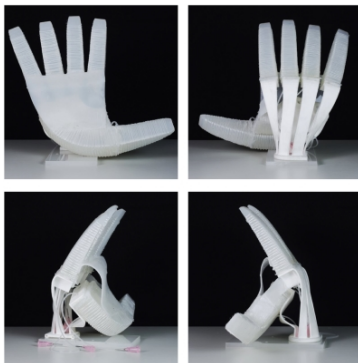
In a gripper, the second type of joint coupling can be used to reduce the number of actuators by removing unnecessary degrees of freedom, while retaining the complex kinematic allowed by a high number of joints. It can be used for example to mimic the

coupled movement of the intermediate and distal phalanges of the human's fingers (as in the Dexmart gripper, shown in Figure 1.9b). It can also be used to create a coupled abduction-adduction motion of several fingers, such as for example in the SARAH gripper [51], where one actuator is used to move in a coupled way two fingers relative to the palm.



(a) Examples of differential mechanisms. θ_a is the input variable, θ_i the output variables, and θ_{a_i} the internal actuation variables. Picture: copyright ©2008, Springer-Verlag Berlin Heidelberg [7]

(b) The SARAH Gripper, an underactuated gripper based on differential mechanisms [51]. Picture: copyright ©2022 Université Laval [52]



(c) The RBO Hand 2 gripper, that implements underactuation through compliant materials [53]. Picture: copyright ©2016, SAGE Publications

(d) The BarrettHand [6], an underactuated gripper using triggered mechanisms. Picture: CC-BY license [54].

Figure 1.12 – Different types of underactuation.

Likewise, underactuated systems are not used only in grippers, and can be divided mainly in three categories: differential mechanisms, compliant mechanisms, and triggered mechanisms [55]. Underactuation not only reduces the number of actuators, but also gives the gripper an adaptive behavior, which makes it able to comply with the object

geometry.

- Differential mechanisms are mechanisms having two degrees of freedom, and which need two inputs to produce an output or conversely, can produce two outputs from a single input. Concretely, they can consist of gears, four bar linkages, seesaws, or special arrangement of pulleys and cables. Examples of such systems are shown in Figure 1.12a. For the specific case of grippers, these mechanisms use springs and mechanical stops to create a defined rest position. For example, the SARAH gripper use such mechanisms, and is displayed in Figure 1.12b.
- Compliant mechanisms use non-rigid bodies, for example soft or elastic materials. For example Deimel and Brock [53] designed a gripper made of silicone rubber and other soft materials, and actuated thanks to compressed air. It is displayed in Figure 1.12c.
- Triggered mechanisms usually use complicated mechanical designs relying on friction to switch the actuation from one output to the other [6, 7]. The mechanical principle used is close to the one used in clutches. An example of gripper using such mechanism is the BarrettHand, displayed in Figure 1.12d.

It is worth noting that this classification is not exhaustive, and that different types of underactuation can coexist in the same system.

In a gripper, underactuation can be found in two different places: in the fingers themselves, and between the fingers [7]. In the fingers, the underactuation allows the phalanges to adapt to the object shape, by distributing the torque between the finger joints: this enables grippers with less actuators than phalanges, and potentially only one actuator by finger. Between the fingers, underactuation allows to dispatch the torque between them: here, the adaptive behavior is extended to the whole gripper. Using underactuation both inside the fingers and between them allows to design a gripper with less actuators than fingers, and even grippers with only one actuator, independently from the number of fingers and phalanges.

Thus, by combining the joint coupling and underactuation mechanisms, one can design grippers able to adapt to various object geometries and to produce robust grasps with very few actuators, while still allowing precision grasps using only the fingertips. For example, the SARAH gripper [51] uses both underactuation and joint coupling, the underactuation being based on differential mechanism, both in the fingers and between them. It has three three-phalanx fingers, with a total of ten degrees of freedom and eleven joints, but is driven by only two actuators. This gripper can perform cylindrical, spherical or planar grasps, for both power and precision grasps.

Multifingered underactuated grippers can be seen as an intermediate mechanical architecture between simple grippers as described in subsection 1.1.2, and complex dexterous grippers as presented in the previous subsection. On one hand, fully actuated multifingered grippers and anthropomorphic grippers aims at achieving versatile and dexterous behavior through bio-mimicry. On the other hand, underactuated grippers aim at achieving versatility through simplicity and mechanical intelligence, and often set dexterity aside.

1.3 Grasp Planning Principles

The versatile ability of a robotic setup does not depend only on the mechanical architecture of the gripper: the chosen grasp planning approach is a determining factor.

1.3.1 Problem statement



Definition 5 (Grasp planning). Process by which the configuration of the gripper (the finger configuration in case of a multifingered gripper) is chosen as well as the approach trajectory allowing to grasp an object. This process takes into account any available information related to the characteristics of the object, of the task to be realized and of the environment.

Humans are able to perform grasp planning effortlessly and instantly: when we need to grasp an object, we immediately know where we should place our hand in an efficient way in relation to the task to be executed, how to position our fingers relative to the object to ensure its stability, how to avoid collisions with our environment when approaching our hand and when handling the object... The functioning of our brain is the legacy of millions of years of evolution, and we spent most of our infancy and childhood learning how to interact with our environment and perfecting our sensorimotor skills. Our ability to plan grasps easily is the result of these two factors.

Replicating this seamless grasp planning process on a robotic system is an open research topic. It can be seen as an optimization problem, that admits input and output data, hypotheses, and constraints.

Input Data Information about the object needs to be gathered: its pose and its physical properties for instance. Then, the task requirements and its environment need to be known too. In addition, the grasp planning process also depends on the gripper properties and abilities.

Output Data Given all these information, a grasp needs to be determined, that is a gripper configuration, as well as its pre-grasp position and approach trajectory.

Hypotheses To reduce the number of unknowns, and thus reduce the number of required sensors, simplifying assumptions can be made on each of the elements required as input data. For instance, knowing the gripper is not a strong hypothesis. Regarding the task requirements and object properties, the grasp planner can be designed for a given target task and target object, for which some hypotheses can be made regarding its properties (weight, friction coefficient,...). Concerning the object pose and environment, it is a stronger hypothesis, but not impracticable in most cases. These hypotheses allow to design a simpler robotic setup and grasp planner, by giving versatility up.

Constraints Independently from the assumptions made, several conditions need to be taken into account when choosing how to grasp the object, in order for the grasp to succeed. From a geometrical point of view, the chosen gripper configuration needs to be kinematically reachable and collision-free with respect to the environment for the gripper-arm system, while, from a control point of view, the grasp needs to ensure object stability and resistance against external perturbations [56]. Thus, grasp planning is simultaneously object dependent, robot hardware dependent, task dependent and environment dependent. Taking into account those constraints to select a gripper configuration is not straightforward, as objects can exhibit sophisticated shapes, gripper-arm combination can have complex or redundant kinematics, environment can be imperfectly known, and the task can have complex and contradictory requirements. The less simplifying assumptions are made, the more complex it becomes.

The versatile grasp planning question is still an open issue and an active research topic [31, 32]. It aims at finding a gripper configuration that allows to grasp an object reliably, with limited a priori knowledge of this object and its environment. The stake is to tend towards the human ability to find efficient grasps for various tasks, even for unseen objects or in an unknown environment.

1.3.2 Overview of Grasp Planning Approaches

A wide variety of grasp planning algorithms exists. They can be differentiated by their level of versatility and complexity, as summarized in Figure 1.13. Most existing methods assume a known task, or consider a generic grasping task with no specific target application. Concerning the gripper, the vast majority of works are made with a given gripper architecture, but some works try to generalize to several gripper architectures [57].

The simplest grasp planners have the most assumptions (approach 1. in Figure 1.13), and were predominantly used before the increasing use of vision systems in robotic cell. It is assumed that the object geometry and position are known, as well as its surrounding environment. The object location is hard-coded in the algorithm, as well as the grasp configuration, which may be found by trial and error when setting up the robotic cell. The grasping process can be triggered after ensuring that the object is at its expected place, with for example optoelectronic sensor. The repeatability of the part positioning before the grasp is of capital importance, as it conditions the validity of the known object location assumption. This can be done by using special carriers or magazines, or special vibrating conveyors that allow the parts to settle into a predictable orientation [1].

A first layer of complexity and machine intelligence can be added by removing the assumption of a known object pose (approach 2. in Figure 1.13). This is typically the case for bin picking tasks: the object geometry and physical properties are known, but it can have any pose in a given area of the workspace. In that case, vision systems, that is a 2D or 3D camera (as shown in Figures 1.3a and 1.3b), or a laser sensor, is needed to locate the object. Typically, the grasp planning process is divided in the following steps [1]:

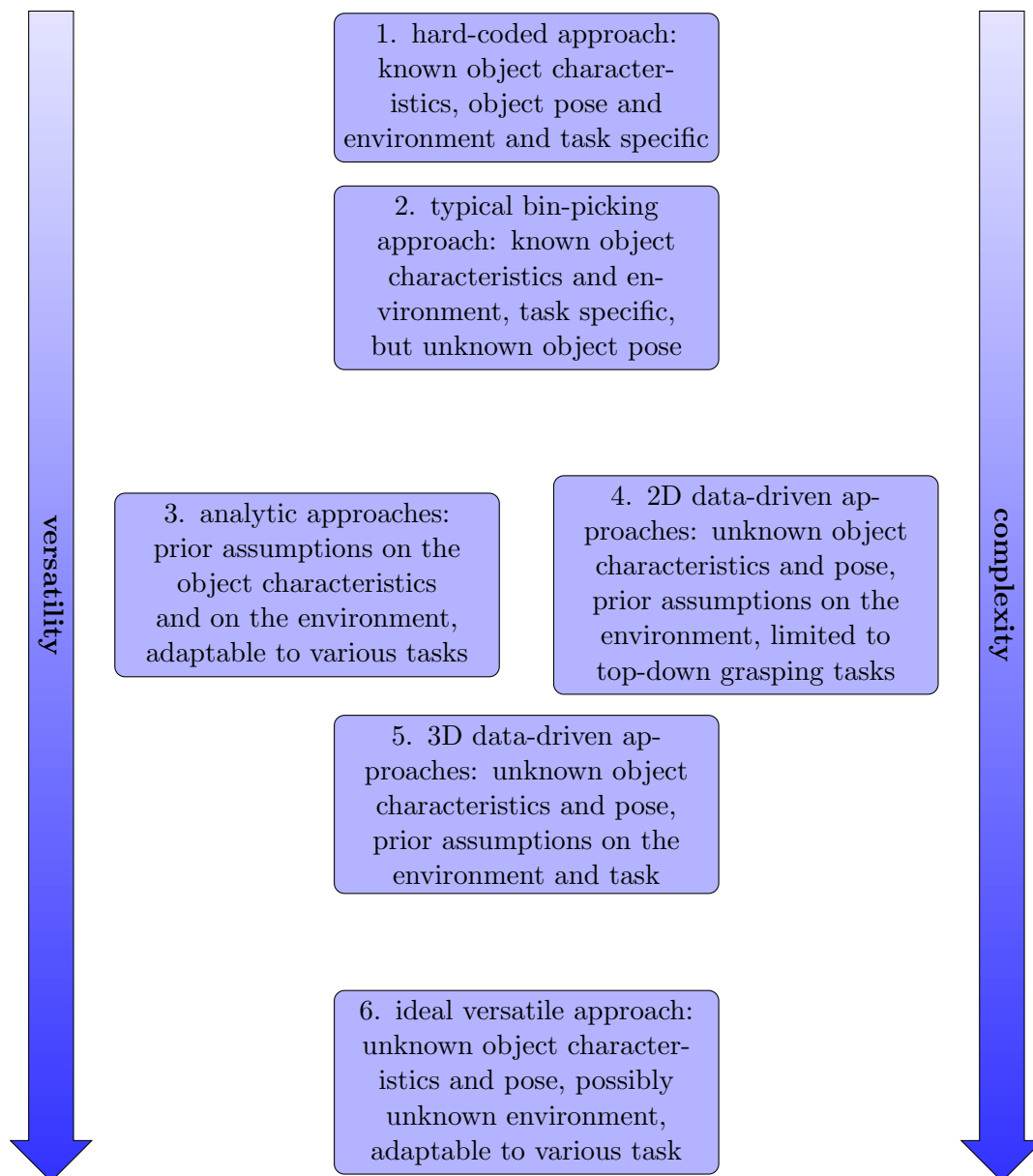


Figure 1.13 – Scheme of grasp planning approaches and their relative complexity and versatility.

1. initial data acquisition: the vision sensor captures a point cloud, an RGB-D (depth-map) or an RGB image.
2. object detection: the goal is to find and isolate in the 2D or 3D input data the known model(s) of the object(s) of interest. This can be done thanks to bounding box or segmentation techniques. Variants exist for both 2D or 3D inputs [31].

3. object pose estimation: this phase aims at determining the pose of the known object in a reference coordinate frame, from the information gathered in the object detection phase. In case of planar grasp, when objects are not expected to be piled on each other, the pose can be expressed as a 2D position in a plane, and a rotation angle relatively to that plane. This can be solved by matching feature points or contour curves [31]. When object can be piled up, a depth information is needed, and the full 6D pose needs to be retrieved. This can be solved by mainly three methods [31]: correspondence-based [58], template-based [59] or voting-based [60].
4. grasp configuration choice: once the object pose is known, a grasp configuration needs to be chosen. As for grasp planners assuming a known object pose, the grasp configuration needs to be specified in the object frame beforehand by an operator [1]. As the object pose was determined in the previous step, a frame transformation allows to express the chosen grasp in the reference frame. Specifying several grasp configurations, in a discrete or continuous manner, can increase the ability to handle various object poses [61], the choice between them being made on a reachability criterion for example.
5. path planning: a collision-free path needs to be found to reach the chosen grasp configuration [1]. For that, the environment needs to be known, or be monitored with vision systems or other sensors.

Finally, grasp planning approaches that do not expect a single known object model, that is approaches 3, 4 and 5 in Figure 1.13, are the most complex to date. Currently in an industrial context, tasks where the object is unknown or frequently changing are still typically performed by a human or with human supervision, and the automation potential is very high. These applications are for example assembly or handling tasks in a small batch or personalized manufacturing context, bin picking tasks with various objects in the bin, or order picking tasks.

In that case, the grasp planner should be able to find autonomously and for any object a configuration that fulfills a given criterion, for example a grasp quality one, as described in section 2.2. There are two main ways to achieve this: analytic approaches (approach 3 in Figure 1.13) and data-driven approaches (approaches 4 and 5 in Figure 1.13) [56]. Both type of approaches have a common structure, with offline and online phases. Their general principle stays the same, and is schematized on Figure 1.14. In particular, they both rely on a grasp space exploration step, explained in further details in section 1.4, this step being the main focus of this thesis. The main principle of analytic and data-driven approaches is described in the following.

- **analytic approaches** (approach 3 in Figure 1.13): those approaches rely on an analytic description of the grasping problem [62–67]. They are often suited for multifingered gripper architectures, which allows these approaches to be applied to various tasks.
 - Offline phase: for a set of objects with known characteristics (geometry, friction, etc...), an analytic quality metric (grasp quality or task-oriented criterion

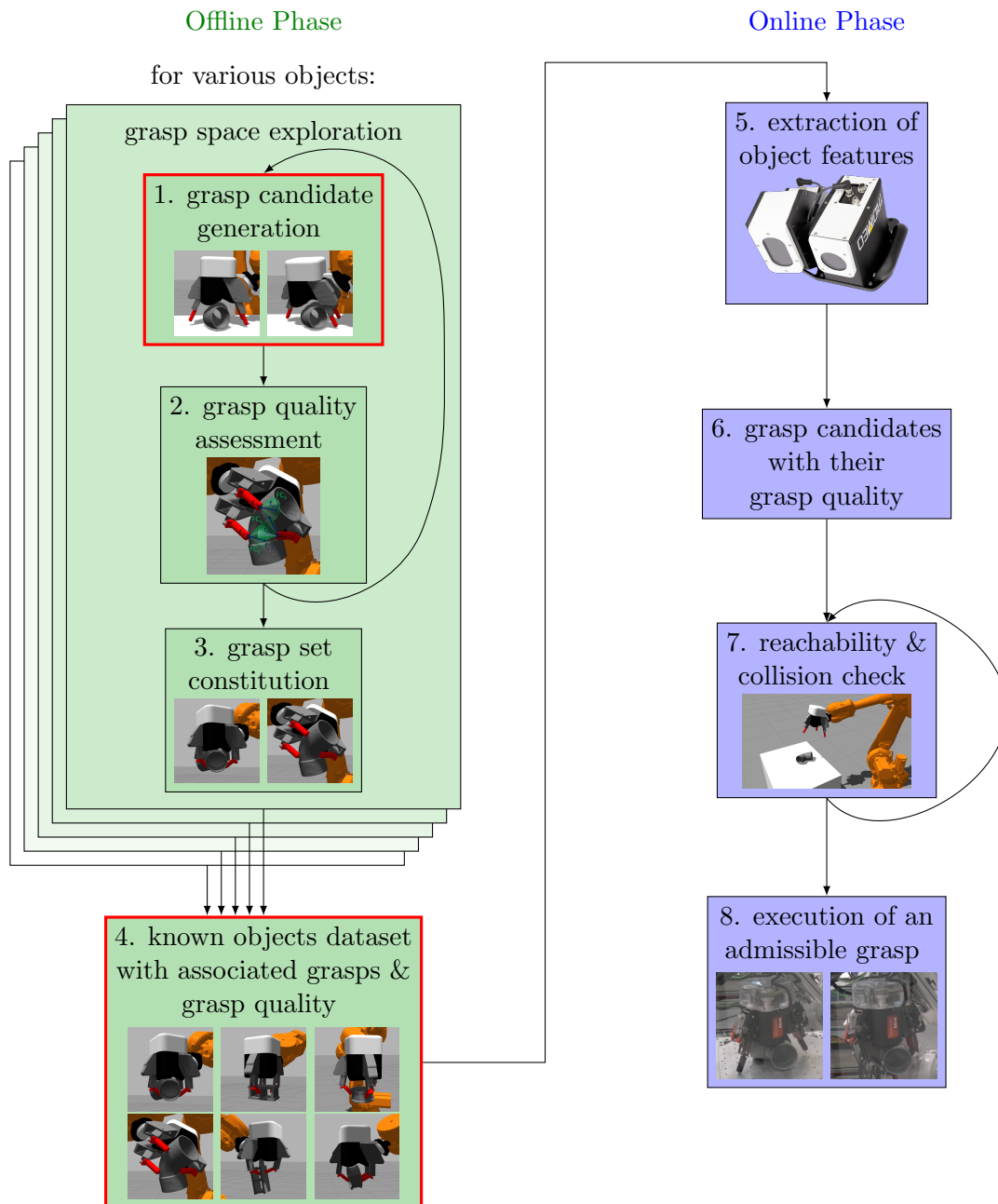


Figure 1.14 – General principle of grasp planners. The main focus and contribution of this thesis is boxed in red.

for instance, that will be further explained in section 2.2) is computed for a set of possible grasps (step 1 and 2 in Figure 1.14). Those possible grasps are found in a grasp space exploration step. This exploration relies on sampling

[62–64], heuristics [65], or optimization techniques [66] for selecting grasp configurations. Finally, grasps with high quality are stored in a dataset (step 3 and 4).

- Online Phase: it is assumed that the object exists in the dataset constituted offline and can be recognized. This recognition can be made using vision system and image processing algorithm [58–60] for example (step 5). The corresponding precomputed grasps are retrieved from the dataset, and ordered by grasp quality (step 6). The best grasp is executed if it is kinematically reachable and collision free (step 7 and 8 in Figure 1.14). Otherwise, an other grasp is selected until a valid one is found.

There exist also methods that do not rely on an explicit dataset of precomputed grasps, but rather analyze the object geometry and directly perform the grasp space exploration online using a set of heuristics, such as in Recatalá et al. [67].

- **data-driven approaches** (approaches 4 and 5 in Figure 1.13): those approaches depend on machine learning methods to predict grasps [57, 68–74]. They are often applied on bi-digital grippers, an architecture choice that limits the versatility of the approach.
 - Offline phase: in a similar way as for the analytic approaches, a grasp dataset for known objects is constituted in a grasp space exploration phase (step 3 and 4 in Figure 1.14). The grasp quality is often an empirical metric [57, 68–71], that is whether the object is successfully grasped or not, but a few approaches using an analytic grasp quality metric have also been reported [72–74] (step 2). Then, from the dataset constituted in step 4, a machine learning algorithm uses object RGB images [70], depth images [68, 69, 74], point clouds [57, 71, 72], or analytic representation (for example superquadrics) [73] as inputs to learn to predict corresponding grasps and grasp quality.
 - Online phase: object features are captured, corresponding to the ones used for the training of the machine learning algorithm (point cloud, RGB image...), thanks to a vision system (step 5). The trained model uses this input to predict grasp candidates with their quality [68, 69, 71–73], or to predict grasp quality for grasps generated by an other method [57, 70, 74] (step 6). If the objects in the training dataset are diversified enough, the learnt model can have decent performances on unknown objects. The rest of the online process is similar to the analytic one (step 7 and 8).

It is worth noting that this general overview is not exhaustive, as many variations around the above description have been developed in the literature.

1.3.3 Limitations

The lack of versatility is the main issue of grasp planners using simplifying hypotheses on the object, the task or the environment.

Grasp planning with hard-coded object characteristics and pose (approach 1 in Figure 1.13) requires a potentially long and tedious tuning phase when setting up the robotic cell to ensure the proper grasping of the part, which have to be done almost from scratch if any component of the grasping task is modified. Moreover, ensuring a reproducible object pose can be challenging or impossible for some object geometries, and in any cases requires specialized carriers or vibrating conveyors that need to be carefully designed, and that may also need a tuning phase. Designing and setting up such process is overall time consuming. This can be profitable only for some specific use cases, mainly for mass-produced parts.

Introduction of vision sensors to allow some variability in the part location can solve some of these limitations (approach 2 in Figure 1.13). Indeed, having a not fully reproducible object pose corresponds to a wide range of possible applications. However, the grasp synthesis, that is the choice of possible grasp in the object frame, is still made beforehand by a human, and will most likely be task specific. It is time consuming, as the specified grasp robustness need to be tested and eventually tuned for each new object, and it may have to be done again if the specified grasp of a given object needs to be changed for a new task.

Methods that tackle the grasp planning issue for unknown objects and unknown poses (approaches 4 and 5 in Figure 1.13) are the most likely to show versatile behavior. However, the grasp planning algorithm is dependent on the gripper architecture considered. A lot of works consider a parallel gripper [57, 68–71, 74], which has its limitations in term of versatility as shown in subsection 1.1.2: the planning algorithm will not be able to overstep them. The main advantage of bi-digital gripper is their simplicity of use. However, this simplicity prevents such grippers to adapt their configuration to the object or to the task. Moreover, they offer a minimal number of contact points, which can be an issue for the grasp robustness. On the contrary, analytic methods (approach 3 in Figure 1.13) use multifingered grippers that are more versatile and adaptable, but these methods are often limited to objects with known characteristics [62–66].

Finally, the actual versatile skill of the grasp planner and gripper architecture combination is conditioned by the grasp space exploration method used to create the grasp dataset (step 1 in Figure 1.14). Indeed, the grasps produced online by the analytic methods are found in this step, and the ones produced by the data-driven methods are generated by a machine learning algorithm that uses as training data the grasps found in this step. Thus, the versatility and success rate of the grasp planner depends directly on the ability of the grasp space exploration to find a wide variety of high quality grasps. The grasp space exploration issue is detailed in the following section.

1.4 Grasp Space Exploration Issue

1.4.1 Definition

An issue shared by all versatile grasp planning methods is the grasp dataset creation, that is the grasp space exploration (step 1 in Figure 1.14). A variety of grasps needs to

be discovered by exploring the space of possible grasp configurations.



recall the definition of grasp space:

Definition 4 (Grasp space). In this work, this concept refers to the set of all possible grasps that a given gripper can produce on a given object. This space is a subset of the gripper configuration set.

The grasp space has two main features that make its exploration a complex issue and a potentially time consuming step: the high dimensionality of the configuration space in which it is embedded, and the high number of constraints it is subject to.

The gripper configuration space, of dimension $d_{conf} = d_{space} + d_{int}$, is constituted of the spacial configuration space of the arm end-effector, of dimension d_{space} , plus the internal configuration space of the gripper itself, of dimension d_{int} . Generally, grasping task are performed in the euclidean space $SE(3)$. For its part, the gripper has an internal configuration space if it has reconfigurable fingers, or if the fingers have multiple actuated phalanges. Actually, the internal configuration space is constituted by every gripper internal degrees of freedom that allow the reconfiguration of fingers in relation to each other. Thus, for a simple parallel gripper, $d_{conf} = 6$, with $d_{space} = 6$ and $d_{int} = 0$. In this case, the internal configuration space has zero dimensions as the only degree of freedom associated with the finger closing does not allow them to change their configuration: the closing trajectory is always the same relative to the palm. The configuration space can reach easily more than a dozen dimensions in case of a multi-fingered hand, due to the increasing size of the internal configuration space.

The grasp space is a subset of this high dimensional gripper configuration space, and it is probably also a set of submanifolds of the gripper configuration space: indeed, for a gripper configuration to belong to the grasp space, it must respect several non linear constraints, that define the topology of the grasp space.

- The configuration is subject to geometric constraints produced by the contacts between the object and the gripper. Indeed, the gripper and the object need to be in contact, but without interpenetrating.
- In case the previous constraint is validated, the produced contacts have to allow the grasp to resist external disturbances to a certain extent. In particular, the grasp needs to resist at least to the object own weight.
- The configuration needs to be compatible with the gripper-arm system kinematics. Actually, for a given robot arm manipulator, some Cartesian poses are unreachable due to kinematics singularities or joint limits. Thus, depending on the object pose relative to the robot base, some part of the grasp space may not be admissible.
- To be able to grasp the object, the configuration needs to avoid collision between the gripper-arm system and the environment, as stated previously. Thus, depending

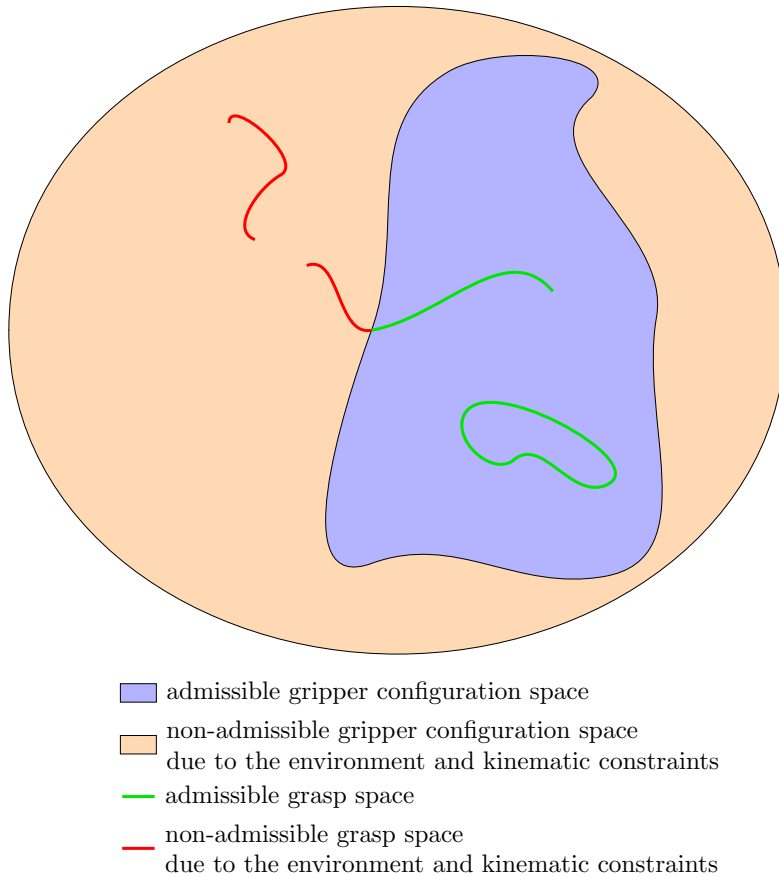


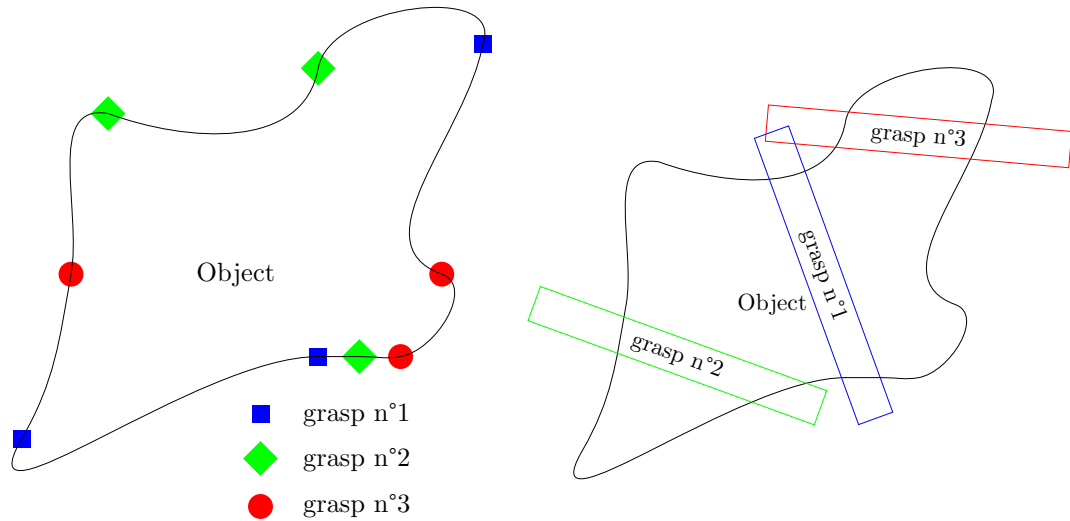
Figure 1.15 – Scheme of an hypothetical two dimensional gripper configuration space, with the grasp space of a given object at a given position in the workspace.

on the environment and on the object pose relative to it, some parts of the grasp space are not admissible.

These constraints make the grasp space relatively small compared to the gripper configuration space. However, it can still be of high dimension, depending on the dimension of the gripper configuration space. A simplified representation of an object grasp space in the hypothetical case of a two dimensional gripper configuration space is shown in Figure 1.15. In a real case, building such representation is extremely complex due to the high dimensionality of the gripper configuration space, and the fact that the nature of the constraints defining the grasp space topology makes it difficult to have an analytic description of it.

There are two main approaches to explore the grasp space, depending on which constraint is given priority [64]:

- Contact point approaches, where the grasp space exploration comes down to test various combinations of contact point locations on the object surface [63, 65, 66,



(a) Scheme of three grasps expressed with the contact point approach on a two dimensional object, in a particular case where the grasps have three contacts each.

(b) Scheme of three grasps expressed with the gripper configuration approach on a two dimensional object, in the particular case of a parallel gripper, where the rectangle small sides represent the jaws, and the wide sides the jaw closing directions.

Figure 1.16 – Scheme of the contact point and gripper configuration approaches for grasp space exploration.

72]. This approach is displayed in Figure 1.16a. Here, the priority is given to compliance with geometric constraints caused by the grasp, as contact points are by definition on the object surface. The main advantage is that it allows to check easily if the chosen contact points can produce a valid grasp and withstand external perturbation. However there is no guarantee that a given combination is a priori kinematically admissible for a given gripper: each contact point requires the inverse kinematics to be computed, to check if the gripper can achieve it. Likewise, if the contact point locations are admissible for the gripper, the required gripper pose may be unreachable for the arm. Moreover, the configuration may not be collision-free with the environment. These constraints need to be tested afterward to ensure that the tested configuration is truly in the grasp space. The main drawback of this approach is the limited number of contact points that can be considered: each contact point requiring two parameters to set its position on the object surface, the parameter space to be explored can become huge.

- Gripper configuration approaches, where the grasp space is explored by testing several gripper spatial configurations and internal configurations, before closing the fingers on the object [57, 68–71, 74]. This approach is shown in Figure 1.16b. One

of the advantages of this approach is that the kinematic reachability and collision-free constraints can be verified more easily. Moreover, being already in the gripper configuration space, this approach is more straightforward than the contact point approach, and requires less inverse kinematics computations. Nevertheless, there is no assurance that a given gripper configuration complies with the geometric and dynamic constraints required to form a grasp: there is no information on the number and locations of the contacts between the gripper and the object when closing the fingers, if any, nor on their ability to resist external forces. In that case, extensive simulation trials need to be realized to check the grasping ability of each configuration. An other drawback is that this approach struggles to deal with multifingered grippers having an internal configuration space of high dimension, due to the increase in the number of parameters to explore.

To circumvent the issue related to the potentially huge size and complexity of the grasp space, numerous contact point approaches limit their search to fingertip contacts [63, 65, 66, 72], and gripper configuration approaches often use a parallel gripper and limit their search to planar grasps [68–70, 74]. Other works using parallel grippers choose to search gripper configurations in the six dimensional euclidean space, but still have to set some constraints for the grasp space exploration. Those constraints can be for example to set the gripper approach vector normal to the object surface [71], or sampling grasps aligned with some point candidates on a point cloud, which prevents grasp generation on unrepresented object areas [57]. Those approaches require a significant amount of trials.

1.4.2 Grasp Space Exploration with Underactuated Grippers

Even if underactuated multifingered grippers are well suited for grasping tasks [6, 7], the grasp space exploration with such grippers is more complex. Indeed, their adaptive underactuated mechanical systems generate both gripper- and object-dependent grasp configurations, as visible in Figure 1.17 with an underactuation based on differential mechanisms.

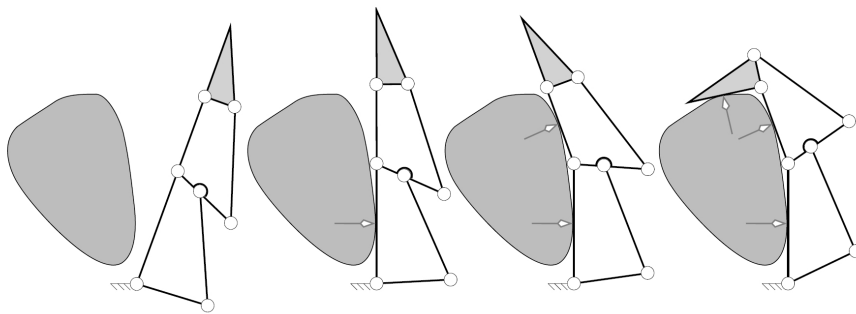


Figure 1.17 – Grasping sequence of an underactuated finger using differential mechanisms [7]. Picture: copyright ©2008, Springer-Verlag Berlin Heidelberg

On one hand, the contact point approaches for grasp exploration described in the

previous subsection (see Figure 1.16b) are inapplicable: the contact point locations are not controllable, as the final finger configuration depends on the geometry of the object and on the final force distribution among fingers and phalanges at the end of the grasp. Indeed, as shown in Figure 1.17, the reaction force at each contact point balance the joint torque transmitted by the actuator. Then, the differential mechanism allows the force to be transmitted to the next joint, until the torque at every joint are balanced by all contact reaction forces.

On the other hand, the gripper configuration approach can become very costly computation wise, due to several limitations.

- The internal configuration space of such grippers can be relatively high, as some fingers can often be reconfigured.
- Due to the high versatility of such architecture, it would be counterproductive to reduce drastically the spatial configuration space, as it is done with the planar grasp hypothesis commonly used with bi-digital parallel grippers.
- The high number of simulation trials generally required by gripper configuration approaches become itself problematic for underactuated multifingered grippers for several reasons.
 - Such grippers have a lot of links, which complicates and slows down the simulation, by increasing the number of body to simulate.
 - In case of underactuation using differential mechanisms, they often include the use of springs, which can cause computational instability in case of explicit or semi-implicit integration, thus requiring to take a smaller timestep.
 - In case of underactuation based on compliant mechanisms, such behavior is even more complex to simulate, and very few simulators are able to handle properly soft and compliant materials.

Thus, there are few works which tackle the grasp planning issue for underactuated multifingered grippers, probably because of the complexity of the grasp space exploration in such cases. Often, some hypotheses restricting the grasp space are chosen, which reduce the versatile potential of the gripper. In Choi et al. [75], the search space has been limited by discretizing it into 24 different possible combinations of approach vectors and wrist orientations. This choice was most likely conditioned by the fact that only real robot trials were conducted, due to the chosen underactuation system, very difficult to simulate accurately because of the presence of compliant materials. In Pelosof et al. [73], a gripper configuration approach is used, and the simulation cost is reduced by considering as object to grasp only superquadrics shapes, which can be simulated very easily as they are described by few parameters. Moreover, the chosen gripper uses a triggered underactuation mechanism, which is simpler to simulate than differential mechanisms using springs.

In some works, a human input is proposed. For example, in Santina et al. [76], a set of ten global grasp primitives has been identified from human examples, and the grasp

space has been reduced to those primitives only for any objects. This highly restricts the grasp space, as it considers that, when grasping any given object, there are only ten possible grasps to choose from, and that one of them is the one and only way to grasp this specific object. This approach does not allow to chose a different grasp depending on the task for example.

1.5 Conclusion

Grasping is a key ability in manufacturing industries. However, grippers currently used in an industrial context are highly specialized, and cannot adapt easily to different tasks or objects, which limits the potential for further automation.

The question of the versatile robotic grasping is still open. To achieve this, two components are needed: a mechanical architecture with sufficient kinematics ability, able to handle various grasping tasks and objects, and a grasp planning algorithm capable of leveraging this mechanical architecture to plan versatile grasps.

Regarding the grasp planning algorithms, state-of-the-art approaches are mainly divided in two categories: analytic and data-driven. Both rely on a grasp space exploration step, that aims at finding various grasps in the potentially large gripper configuration space, and evaluate their quality, often offline through simulation. To simplify this exploration, assumptions are often made, such as planar grasps or fingertip contacts, which limit the versatility of the approach. Moreover, the overall versatility is also determined by the chosen gripper architecture: a bi-digital parallel gripper is inherently less versatile than a multifingered one with numerous degrees of freedom.

Underactuated multifingered grippers are a powerful tool to achieve versatile robotic grasping. Their mechanical intelligence allows them to adapt passively to a wide variety of objects without the need for numerous actuators and complex controller. However, the grasp planning for this type of grippers is complex, mainly due to the computational cost of the required grasp space exploration step. Indeed, such grippers admit a high dimensional configuration space, which increases the number of parameters to be explored. Their underactuation mechanism is also more complex to simulate than classic grippers.

The main contribution of this thesis is to propose an object dependent efficient grasp space exploration method adapted for underactuated multifingered grippers, based on a set of human suggested grasps, thus leveraging our ability to find efficient and versatile grasping strategies. This method also uses an analytic grasp quality criterion to rank the various grasps generated during the exploration.

Chapter 2

Grasp Modeling Framework

The existing literature regarding grasp analysis is mainly divided in two categories. Some authors focus on extracting grasp and gripper properties for fully actuated multifingered grippers [37, 77–79]. Other authors focus on describing and analysing the specific adaptive behavior of an underactuated finger [7, 80–82], but does not study the grasp or gripper as a whole. A few works at the crossroad between these two categories concentrate on the study of grippers having coupled joints [83, 84]. The contribution of this chapter is methodological: the theoretical knowledge required for this work regarding multifingered grippers and grasp modeling is given, with a focus on the description of the influence of underactuation on grasp and gripper properties. First, modeling principles for grasp and gripper description are introduced, and the underactuation specificity is highlighted. Then, mathematical tools for grasp characterisation are described, together with an example showing the effect of underactuation. Finally, some relevant state-of-the-art grasp quality metrics are presented, along with the rationale behind the metric choice in our planner framework.

Contents

2.1	Grasp and Gripper Description	54
2.1.1	Contact Description	54
2.1.2	Kinetostatic Modeling of Multifingered Gripper	57
2.1.3	gripper Jacobian Matrix	61
2.1.4	Effect of Underactuation	63
2.1.5	Grasp Map	67
2.2	Grasp Characterisation	70
2.2.1	Grasp Classifications & Desirable Properties	70
2.2.2	Overview of Existing Grasp Quality Metrics	82
2.2.3	Chosen Grasp Quality Metric	87
2.3	Conclusion	88

2.1 Grasp and Gripper Description

Multifingered grippers have tree-like mechanical architecture, made of several polyarticulated serial chains called fingers. The modeling of their behavior when grasping objects is explained in the following.



Definition 6 (Grasp). Refers to the seizure of a given object with a given gripper, according to given physical and environmental conditions. The result is a set of contact points between the object and the gripper along with mathematical and physical properties associated with the produced grip. The contact points are modeled by adopting some hypotheses on the transmitted forces and velocities.

First, the main contact models are presented. The kinetostatic modeling of multifingered gripper is recalled in the fully-actuated case. Then, the influence of the underactuation on this modeling is highlighted. Finally, the Grasp Map formalism is recalled.

2.1.1 Contact Description

Contacts play an important role in grasping as every motion or effort transmitted from the gripper to the object go through them. Depending on the chosen model, different wrench or twist components are transmissible from the contact to the object [37, 77]. Thus, the contact model choice influences the theoretical grasping and manipulation ability of a given gripper. Here, it is assumed a grasping task, with no in-hand manipulation, i.e. the contacts cannot roll or slide, and stay at a given location. A broad presentation of existing contact models can be found in Kao et al. [85] for example. In the following, a non-exhaustive introduction of the three mainly used punctual contact models is given, in the case of a 3D contact.

- The simplest model is the frictionless point contact. This model is obtained when friction between the object and the finger is not considered. In this model, only the contact force in the contact normal direction is transmitted. The others contact wrench components are neglected. The number of transmitted wrench components at the contact is noted n_λ , and in this case $n_\lambda = 1$. Such contact needs to enforce the following unilateral constraint:

$$f_n \geq 0, f_n \in \mathbb{R} \quad (2.1)$$

with f_n the normal force component of the contact wrench. This type of contact occurs rarely in practice: it can model accurately only a situation where the object surface is slippery, and where the contact area is very small. Still, the simplicity of this model is an advantage. Moreover, using this model is conservative in term of

grasp stability, as the real contact most likely benefits from the additional contact friction that occurs in reality.

- For its part, the point contact with friction model admits the transmission of tangential forces, so that every translational force wrench components can be applied at the contact point. The moments are considered null. In this case, $n_\lambda = 3$. For this type of contact, the following constraints need to be respected, in the classic case of a Coulomb friction model:

$$\sqrt{f_{tx}^2 + f_{ty}^2} \leq \mu f_n, f_n \geq 0 \mid (f_{tx}, f_{ty}, f_n) \in \mathbb{R}^3 \quad (2.2)$$

with f_{tx} and f_{ty} the tangential force components of the contact wrench, and $\mu > 0$ the friction coefficient, an empirical parameter which is function of the materials of the object and the finger. This constraint can be represented geometrically as a cone having its height along the contact normal, where the set of applied contact forces must lie. The point contact with friction model corresponds to a practical case where contact friction exists and cannot be neglected, but the contact surface is sufficiently small to neglect friction moment.

- The last model is the soft-finger contact. In this model, in addition to the wrench components transmitted by the point contact with friction model, the moment about the contact normal is also transmitted. The moments about the two other axes are still considered null. For a contact enforcing this model, $n_\lambda = 4$. A contact modeled this way has to enforce the constraints:

$$\sqrt{f_{tx}^2 + f_{ty}^2} \leq \mu f_n, |\tau_n| \leq \gamma f_n, f_n \geq 0, \mid (f_{tx}, f_{ty}, f_n, \tau_n) \in \mathbb{R}^4 \quad (2.3)$$

with τ_n the normal moment component of the contact wrench, and $\gamma > 0$ the coefficient of torsional friction, which is, as μ , an empirical parameter depending on the materials of the surface involved in the contact. This type of contact corresponds to situations where the contact area is large, which allows the transmission of a moment along the contact normal. Such situations are in practice very common, that is why this model, despite being more complex, is also more realistic. However, this level of realism is not mandatory for all use case.

The main features of the presented contact models are summarized on Table 2.1. The constraints on transmitted wrenches also hold for twists.

The wrench or twist effectively transmitted by the contact point $\mathbf{w} \in \mathbb{R}^{n_\lambda}$ (respectively $\boldsymbol{\xi} \in \mathbb{R}^{n_\lambda}$) can be expressed from $\tilde{\mathbf{w}} \in \mathbb{R}^6$ (respectively $\tilde{\boldsymbol{\xi}} \in \mathbb{R}^6$) the contact wrench (or twist) applied at the contact point and the matrix $\mathbf{H} \in \mathbb{R}^{n_\lambda \times 6}$ as follows:

$$\mathbf{w} = \mathbf{H}\tilde{\mathbf{w}} \qquad \boldsymbol{\xi} = \mathbf{H}\tilde{\boldsymbol{\xi}} \quad (2.4)$$

The total wrench or twist effectively transmitted by a set of n_c contacts (for example during a grasp) is similarly computed with \mathbf{H}_g a block matrix. With $n_\lambda = n_{\lambda 1} + \dots + n_{\lambda n_c}$,

\mathbf{H}_g is expressed as:

$$\mathbf{H}_g = \begin{bmatrix} \mathbf{H}_1 & & & 0 \\ & \mathbf{H}_2 & & \\ & & \ddots & \\ 0 & & & \mathbf{H}_{n_c} \end{bmatrix} \in \mathbb{R}^{n_\lambda \times 6n_c} \quad (2.5)$$

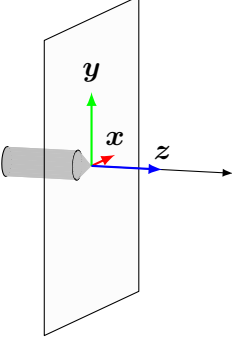
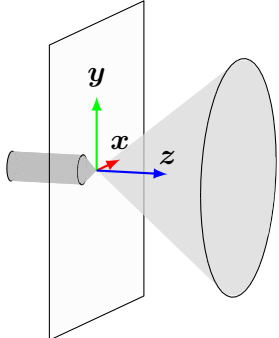
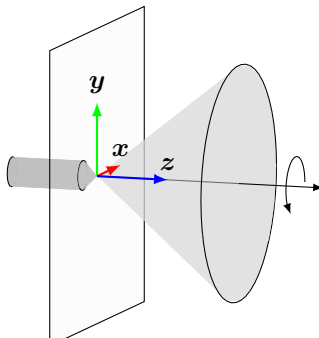
Contact types	Scheme	corresponding \mathbf{H} matrix	Constraints
Frictionless point contact		$\begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}$	$f_n \geq 0$
Point contact with friction		$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}$	$\sqrt{f_{tx}^2 + f_{ty}^2} \leq \mu f$ $f_n \geq 0$
soft-finger contact		$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$	$\sqrt{f_{tx}^2 + f_{ty}^2} \leq \mu f$ $f_n \geq 0$ $ \tau_n \leq \gamma f$

Table 2.1 – Summary of common contact types (in the case of a 3D contact). The contact wrench is expressed in the contact frame, its z axis being toward the contact normal.

2.1.2 Kinetostatic Modeling of Multifingered Gripper

The framework for finger and gripper modeling is given in the following, in the generic case of a fully-actuated finger.

Finger Description

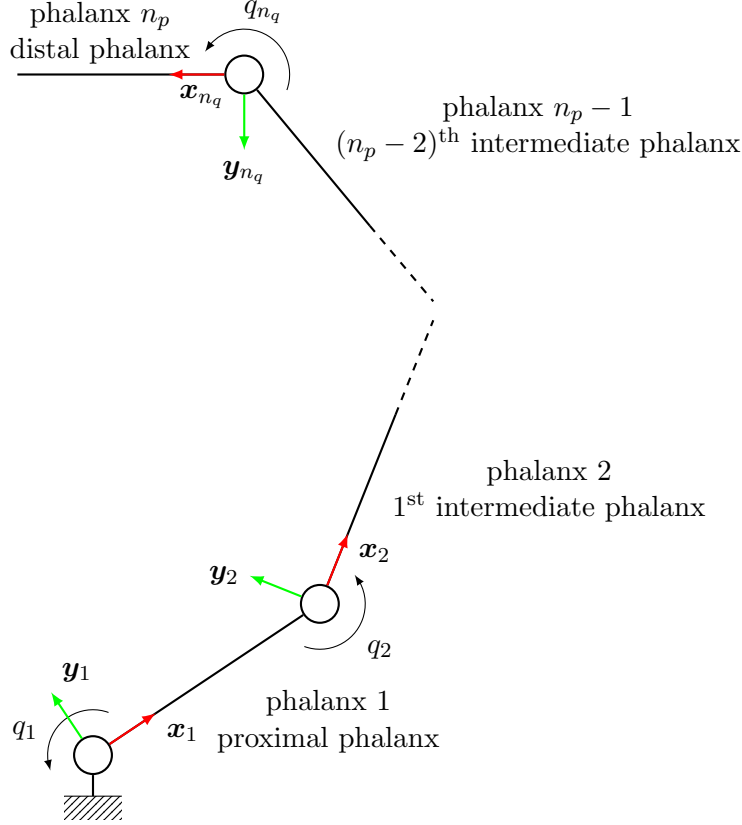


Figure 2.1 – Geometric and kinematic description of a planar robotic finger, with all joints having the same rotation axis. In this particular case, $n_p = n_q$.

A finger is a polyarticulated serial chain, with n_p rigid links called phalanges, and n_q joints. These joints are commonly revolute, but can also be prismatic. n_q is also the number of degrees of freedom of the finger if there is no mechanical coupling. For a fully actuated finger, the finger has also n_q degrees of actuation. The vector of joint angles is denoted by \mathbf{q} :

$$\mathbf{q} = (q_1, \dots, q_{n_q})^T \in \times_{i=1}^{n_q} [q_{i,\min}, q_{i,\max}] \quad (2.6)$$

with $q_{i,\min}$ and $q_{i,\max}$ respectively the minimum and maximum joint limits of the i^{th}

joint. The input joint torque vector is noted \mathbf{t} :

$$\mathbf{t} = (t_1, \dots, t_{n_q})^\top \in \mathbb{R}^{n_q} \quad (2.7)$$

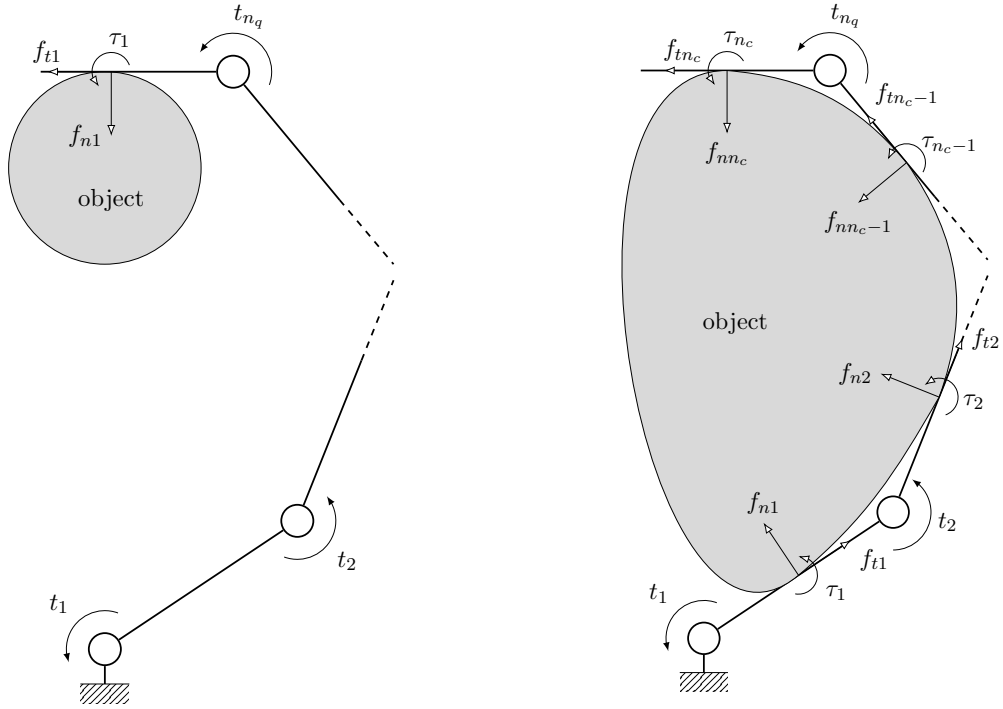
The geometry and kinematics of a planar finger with all joints having the same rotation axis is shown in Figure 2.1. Often, some phalanges can have an additional joint with its rotation axis along \mathbf{y} for an abduction-adduction motion.

When the finger is grasping an object, a set of contact points between the object and the gripper is formed. n_c contacts are involved in a power grasp, and only one at the distal phalanx in a precision grasp ($n_c = 1$).

The wrench and twist applied by a phalanx at the i^{th} contact expressed in the local contact frame are noted $\tilde{\mathbf{w}}_i$ and $\tilde{\boldsymbol{\xi}}_i$, and the vectors of every contact wrench and twist components created by the finger are noted $\tilde{\mathbf{w}}_f$ and $\tilde{\boldsymbol{\xi}}_f$. These wrenches and twists are summarized in Table 2.2. All the wrench and twist components of $\tilde{\mathbf{w}}_i$ and $\tilde{\boldsymbol{\xi}}_i$ may not be transmitted to the object depending on the chosen contact model, as discussed in subsection 2.1.1.

$\tilde{\mathbf{w}}_f$ and $\tilde{\boldsymbol{\xi}}_f$ can be defined as:

$$\tilde{\mathbf{w}}_f = (\tilde{\mathbf{w}}_1^\top, \tilde{\mathbf{w}}_2^\top, \dots, \tilde{\mathbf{w}}_{n_c}^\top)^\top \quad \tilde{\boldsymbol{\xi}}_f = (\tilde{\boldsymbol{\xi}}_1^\top, \tilde{\boldsymbol{\xi}}_2^\top, \dots, \tilde{\boldsymbol{\xi}}_{n_c}^\top)^\top \quad (2.8)$$



(a) Scheme of a finger performing a precision grasp. (b) Scheme of a finger performing a power grasp.

Figure 2.2 – Effort applied by a finger during a grasp in the planar case.

The Figure 2.2 shows an example of the joint torques and contact wrenches on a finger executing a precision grasp and a power grasp in the planar case.

Planning a grasp with a finger requires first to find contact point positions allowing to perform the task at hand. Then, it requires to determine the joint position that allows to reach these contact point positions. To be maintained, sufficient wrenches need to be applied on the contact points, these wrenches being controlled by the joint torques. The relation between finger joint torques \mathbf{t} and contact wrenches $\tilde{\mathbf{w}}_f$ is detailed in subsection 2.1.3.

definition	notation	2D	3D
contact wrench	$\tilde{\mathbf{w}}_i$	$(f_{ti}, f_{ni}, \tau_i)^\top$	$(f_{txi}, f_{tyi}, f_{ni}, \tau_{xi}, \tau_{yi}, \tau_{ni})^\top$
contact twist	$\tilde{\boldsymbol{\xi}}_i$	$(v_{xi}, v_{yi}, \omega_i)^\top$	$(v_{xi}, v_{yi}, v_{zi}, \omega_{xi}, \omega_{yi}, \omega_{zi})^\top$
finger contact wrench	$\tilde{\mathbf{w}}_f$	$\in \mathbb{R}^{3n_c}$	$\in \mathbb{R}^{6n_c}$
finger contact twist	$\tilde{\boldsymbol{\xi}}_f$	$\in \mathbb{R}^{3n_c}$	$\in \mathbb{R}^{6n_c}$
gripper contact wrench	$\tilde{\mathbf{w}}_g$	$\in \mathbb{R}^{3n_c}$	$\in \mathbb{R}^{6n_c}$
gripper contact twist	$\tilde{\boldsymbol{\xi}}_g$	$\in \mathbb{R}^{3n_c}$	$\in \mathbb{R}^{6n_c}$

Table 2.2 – Summary of the contact wrench and twist vectors in the 2D and 3D cases.

gripper Description

A multifingered gripper can be modeled as a tree-like structure, composed of n_f fingers ($n_f \geq 2$), fixed on a common mechanical base, called the palm in the following. In several existing multifingered grippers, some degrees of freedom between the palm and the proximal phalanx of each finger allow to reposition it relative to the palm. These repositioning degrees of freedom can be one or several revolute joints with their axis normal to the axes of the other joints, or one or several prismatic joints, or both. The displacements and input torques of the supplementary degrees of freedom corresponding to the i^{th} finger are included respectively in \mathbf{q}^i and in \mathbf{t}^i . Let n_{qg} be the total number of degrees of freedom of the gripper. If the gripper has n_f identical fingers (that is having the same number of degrees of freedom n_q), then $n_{qg} = n_f \cdot n_q$. The vector of the whole gripper joint displacements is noted $\mathbf{q}_g \in \mathbb{R}^{n_{qg}}$. Likewise, the vector of gripper input joint torques is noted $\mathbf{t}_g \in \mathbb{R}^{n_{qg}}$.

An example of a gripper having three two-phalanx fingers, with an additional revolute joint at the base of each finger is displayed on Figure 2.3. In this example for the i^{th} finger:

$$\mathbf{q}^i = (q_1^i, q_2^i, q_3^i)^\top \quad \mathbf{t}^i = (t_1^i, t_2^i, t_3^i)^\top \quad (2.9)$$

and:

$$\mathbf{q}_g = \left(\mathbf{q}^{1\top}, \mathbf{q}^{2\top}, \mathbf{q}^{3\top} \right)^\top \in \mathbb{R}^9 \quad \mathbf{t}_g = \left(\mathbf{t}^{1\top}, \mathbf{t}^{2\top}, \mathbf{t}^{3\top} \right)^\top \in \mathbb{R}^9 \quad (2.10)$$

For clarity, the object, the contacts, and their associated wrenches are not displayed in Figure 2.3, but the principle stay the same as for a single finger. With n_c contacts on the whole gripper system, the vector of every contact wrench components created by the gripper is noted $\tilde{\mathbf{w}}_g$, and similarly for the twist components $\tilde{\boldsymbol{\xi}}_g$. In case of a precision grasp, there is a contact on each distal phalanx, thus $n_c = n_f$. The gripper contact wrench and twist dimension are summarized in Table 2.2.

When grasping an object, the system constituted of the object and the gripper becomes a parallel kinematic chain. Thus, the joint configuration of each finger becomes subject to kinematic constraints which depend on the object geometry and also on the joint configuration of the other fingers. Not fulfilling these constraints can mean that the grasp is lost, for example if one of the finger is not anymore in contact with the object.

In the following subsection, the existing relationships between gripper joint displacements \mathbf{q}_g , gripper joint torques \mathbf{t}_g , contact twists $\tilde{\boldsymbol{\xi}}_g$ and contact wrenches $\tilde{\mathbf{w}}_g$ are explained.

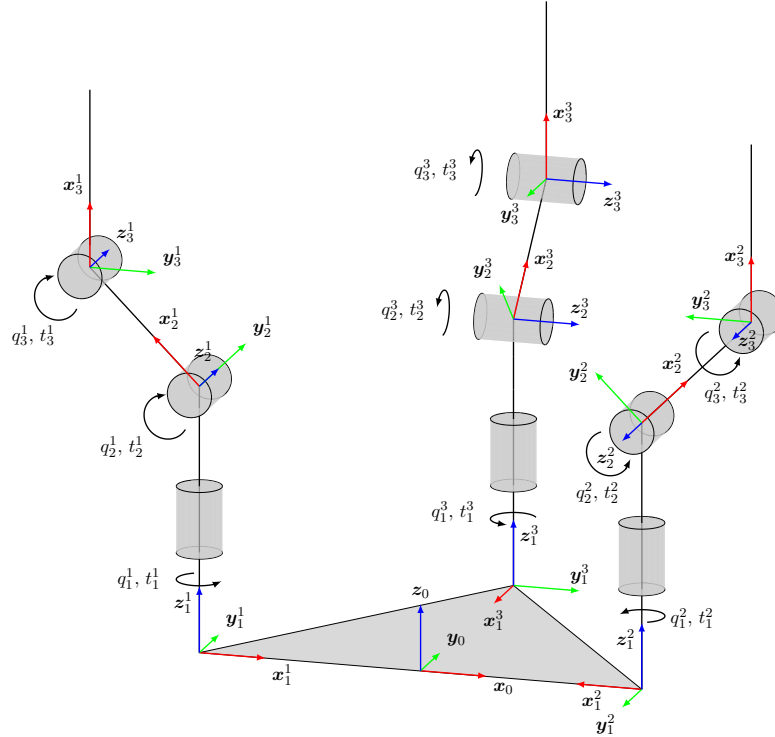


Figure 2.3 – Kinematic scheme of a robotic gripper with three two-phalanx fingers.

2.1.3 gripper Jacobian Matrix

To begin with, the relations between the input joint torques, joint angles and contact wrench are described in the case of a single fully-actuated finger executing a precision grasp in the planar case, as displayed in Figure 2.4. This simple case can then be extended to the 3D case, or to the more complex case of a finger power grasp, or also to the case of a full gripper precision or power grasp. More in depth mathematical development can be found in Birglen et al. [7], Prattichizzo and Trinkle [37], and Murray et al. [77].

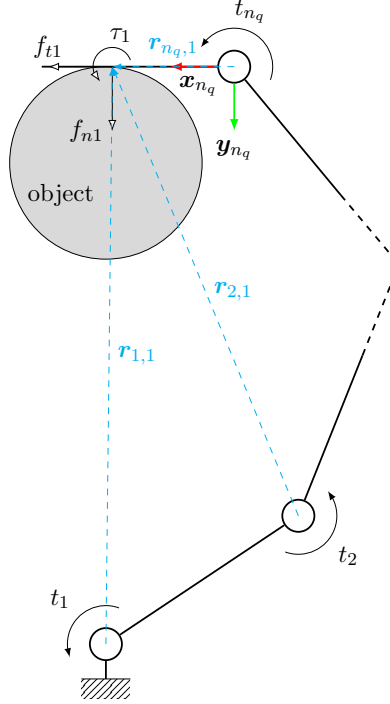


Figure 2.4 – Scheme of a finger performing a precision grasp in the planar case.

First, equating the joint power and the contact power gives:

$$\mathbf{t}^\top \dot{\mathbf{q}} = \tilde{\boldsymbol{\xi}}_1 \odot \tilde{\mathbf{w}}_1 \quad (2.11)$$

with $\tilde{\boldsymbol{\xi}}_1$ and $\tilde{\mathbf{w}}_1$ the twist and the wrench of the contact, and \odot the reciprocal product of screw. The components of \mathbf{t} and \mathbf{w}_1 are displayed in Figure 2.4. $\tilde{\boldsymbol{\xi}}_1$ can be expressed more precisely as follows:

$$\tilde{\boldsymbol{\xi}}_1 = \sum_{k=1}^{n_q} \dot{q}_k \tilde{\boldsymbol{\xi}}_1^k \quad (2.12)$$

with $\tilde{\boldsymbol{\xi}}_1^k$ the joint twist of the k^{th} joint with respect to the contact point. As each joint is a revolute joint, one can express it as follows, with $\mathbf{r}_{k,1}$ the vector from the k^{th} joint

rotation center to the contact point:

$$\tilde{\boldsymbol{\xi}}_1^k = \begin{bmatrix} -\mathbf{r}_{k,1}^\top \mathbf{y}_{n_q} \\ \mathbf{r}_{k,1}^\top \mathbf{x}_{n_q} \\ 1 \end{bmatrix} \quad (2.13)$$

Thus,

$$\mathbf{t}^\top \dot{\mathbf{q}} = \sum_{k=1}^{n_q} \dot{q}_k \begin{bmatrix} -\mathbf{r}_{k,1}^\top \mathbf{y}_{n_q} \\ \mathbf{r}_{k,1}^\top \mathbf{x}_{n_q} \\ 1 \end{bmatrix} \odot \tilde{\mathbf{w}}_1 \quad (2.14)$$

with $\tilde{\mathbf{w}}_1 = \tilde{\mathbf{w}}_f$ in case of a single contact, and by removing $\dot{\mathbf{q}}$ on both sides of the equation, one obtains:

$$\mathbf{t}^\top = \tilde{\mathbf{w}}_f^\top \tilde{\mathbf{J}}_f \quad (2.15)$$

with

$$\tilde{\mathbf{J}}_f = \begin{bmatrix} -\mathbf{r}_{1,1}^\top \mathbf{y}_{n_q} & -\mathbf{r}_{2,1}^\top \mathbf{y}_{n_q} & \cdots & -\mathbf{r}_{n_q,1}^\top \mathbf{y}_{n_q} \\ \mathbf{r}_{1,1}^\top \mathbf{x}_{n_q} & \mathbf{r}_{2,1}^\top \mathbf{x}_{n_q} & \cdots & \mathbf{r}_{n_q,1}^\top \mathbf{x}_{n_q} \\ 1 & 1 & \cdots & 1 \end{bmatrix} \in \mathbb{R}^{3 \times n_q} \quad (2.16)$$

$\tilde{\mathbf{J}}_f$ is called the finger Jacobian. It depends on the geometry of the finger, that is the length of the phalanges, on the position of the contact on the distal phalanx, and on \mathbf{q} , the finger joint angles. It allows to relate the joint input torques \mathbf{t} to the wrench applied by the finger on the contact point $\tilde{\mathbf{w}}_f$. By reasoning in terms of finger contact twist $\tilde{\boldsymbol{\xi}}_f$ (with $\tilde{\boldsymbol{\xi}}_f = \tilde{\boldsymbol{\xi}}_1$ in case of a single contact) and joint velocities $\dot{\mathbf{q}}$, a similar relation involving also this Jacobian matrix can be found [37, 77]:

$$\tilde{\boldsymbol{\xi}}_f = \tilde{\mathbf{J}}_f \dot{\mathbf{q}} \quad (2.17)$$

In case of a finger performing a power grasp with n_c contacts, the relation still holds, but the column dimension of $\tilde{\mathbf{J}}_f$ changes to take into account the n_c contacts, as summarized in Table 2.3. In that case, $\tilde{\mathbf{J}}_f$ depends also on the position of each contact on the phalanges, in addition to the other parameters. This case is studied in depth in Birglen et al. [7].

In order to take into account the contact model and the constraints on transmissible contact wrench and twist component, the finger Jacobian can be redefined as follows, using Equations (2.4) and (2.5), with n_λ the total number of transmitted twist or wrench components:

$$\mathbf{J}_f = \mathbf{H}_g \tilde{\mathbf{J}}_f \in \mathbb{R}^{n_\lambda \times n_q} \quad (2.18)$$

For a gripper with n_f fingers and n_{qg} degrees of freedom performing a grasp with n_c contacts, the relations become:

$$\mathbf{t}_g^\top = \tilde{\mathbf{w}}_g^\top \tilde{\mathbf{J}}_g \quad (2.19)$$

$$\tilde{\boldsymbol{\xi}}_g = \tilde{\mathbf{J}}_g \dot{\mathbf{q}}_g \quad (2.20)$$

with $\tilde{\mathbf{J}}_g$ a block matrix defined as follows:

$$\tilde{\mathbf{J}}_g = \begin{bmatrix} \tilde{\mathbf{J}}_{f1} & & & 0 \\ & \tilde{\mathbf{J}}_{f2} & & \\ & & \ddots & \\ 0 & & & \tilde{\mathbf{J}}_{fn_f} \end{bmatrix} \quad (2.21)$$

This case is developed thoroughly in Prattichizzo and Trinkle [37] and Murray et al. [77]. Table 2.3 summarize the dimensions of the gripper and finger Jacobian matrices. The above gripper Jacobian can also take into account the contact models as follows:

$$\mathbf{J}_g = \mathbf{H}_g \tilde{\mathbf{J}}_g \in \mathbb{R}^{n_\lambda \times n_{qg}} \quad (2.22)$$

The rows of \mathbf{J}_g represent the gripper joint velocity contributions to each transmitted contact twist components. In turn, each row of \mathbf{J}_g^\top corresponds to the transmitted contact wrench component contributions to each gripper joint torque.

definition	notation	2D	3D
finger Jacobian matrix	$\tilde{\mathbf{J}}_f$	$\in \mathbb{R}^{3n_c \times n_q}$	$\in \mathbb{R}^{6n_c \times n_q}$
	\mathbf{J}_f	$\in \mathbb{R}^{n_\lambda \times n_q}$	$\in \mathbb{R}^{n_\lambda \times n_q}$
gripper Jacobian matrix	$\tilde{\mathbf{J}}_g$	$\in \mathbb{R}^{3n_c \times n_{qg}}$	$\in \mathbb{R}^{6n_c \times n_{qg}}$
	\mathbf{J}_g	$\in \mathbb{R}^{n_\lambda \times n_{qg}}$	$\in \mathbb{R}^{n_\lambda \times n_{qg}}$

Table 2.3 – Summary of the Jacobian matrices in the 2D and 3D cases.

2.1.4 Effect of Underactuation

In the following is studied the influence of underactuation on the previous relationships between the quantities associated to contacts and joints. This theoretical study is conducted in the particular case of underactuation based on differential mechanisms as defined in subsection 1.2.3. This specific type of underactuation is chosen because it corresponds to the underactuation system of the gripper used in the experimental setup in chapter 4. In the rest of the manuscript, the term underactuation refers to differential-based mechanisms, unless stated otherwise.

For an underactuated finger using a differential mechanism, one can still relate the joint torques and displacements to the contact wrenches and twists. However, all the joint torques are no longer controllable. In a finger, only some joints are still directly linked to an actuator. For the other joints, the torques are not controllable, and correspond to the elastic forces exerted by the underactuation mechanism. Regarding the joint positions and velocities, none are directly controllable, only the actuators positions and velocities are. They are related to every joint positions and velocities through the underactuation system. These properties and their consequences are explained in the

joints not directly linked to the actuator, $\dot{\mathbf{q}}_j$:

$$\boldsymbol{\omega}_a = (\dot{q}_a, \dot{q}_2, \dots, \dot{q}_{n_q})^\top = (\dot{q}_a, \dot{\mathbf{q}}_j^\top)^\top \in \mathbb{R}^{n_q} \quad (2.24)$$

with

$$\begin{aligned} \dot{\mathbf{q}} &= \mathbf{T}_f \boldsymbol{\omega}_a \\ &= \begin{bmatrix} X_1 & X_2 & X_3 & \cdots & X_{n_q} \\ & 1 & & & 0 \\ & & 1 & & \\ & & & \ddots & \\ 0 & & & & 1 \end{bmatrix} \boldsymbol{\omega}_a \end{aligned} \quad (2.25)$$

$\mathbf{T}_f \in \mathbb{R}^{n_q \times n_q}$ is the finger transmission matrix. This matrix is function of the transmission mechanism used to propagate the input actuation torque and displacement to the i^{th} joint. Each coefficient X_i depends on the specific geometry of the transmission mechanism, and on the joint positions \mathbf{q} . The expression of the coefficient X_i of this matrix is further developed in Birglen et al. [7]. This matrix describes how \dot{q}_a (and indirectly also q_a) is related to the joint velocities and positions by the underactuation mechanism.

Thus, for a single underactuated finger involved in a planar precision grasp, Equation 2.14 becomes:

$$\mathbf{t}^\top \boldsymbol{\omega}_a = \sum_{k=1}^{n_q} \dot{q}_k \begin{bmatrix} -\mathbf{r}_{k,1}^\top \mathbf{y}_{n_q} \\ \mathbf{r}_{k,1}^\top \mathbf{x}_{n_q} \\ 1 \end{bmatrix} \odot \tilde{\mathbf{w}}_1 \quad (2.26)$$

and Equation 2.15 becomes:

$$\begin{aligned} \mathbf{t}^\top \boldsymbol{\omega}_a &= \tilde{\mathbf{w}}_f^\top (\tilde{\mathbf{J}}_f \dot{\mathbf{q}}) \\ &= \tilde{\mathbf{w}}_f^\top (\tilde{\mathbf{J}}_f \mathbf{T}_f \boldsymbol{\omega}_a) \end{aligned} \quad (2.27)$$

finally, by removing $\boldsymbol{\omega}_a$ on both sides:

$$\mathbf{t}^\top = \tilde{\mathbf{w}}_f^\top (\tilde{\mathbf{J}}_f \mathbf{T}_f) \quad (2.28)$$

and by identification from Equations (2.17) and (2.25):

$$\tilde{\boldsymbol{\xi}}_f = (\tilde{\mathbf{J}}_f \mathbf{T}_f) \boldsymbol{\omega}_a \quad (2.29)$$

From Equations (2.16) and (2.25), one obtains:

$$\begin{aligned} \tilde{\mathbf{J}}_f \mathbf{T}_f &= \\ &= \begin{bmatrix} -X_1 \mathbf{r}_{1,1}^\top \mathbf{y}_{n_q} & -X_2 \mathbf{r}_{1,1}^\top \mathbf{y}_{n_q} - \mathbf{r}_{2,1}^\top \mathbf{y}_{n_q} & \cdots & -X_{n_q} \mathbf{r}_{1,1}^\top \mathbf{y}_{n_q} - \mathbf{r}_{n_q,1}^\top \mathbf{y}_{n_q} \\ X_1 \mathbf{r}_{1,1}^\top \mathbf{x}_{n_q} & X_2 \mathbf{r}_{1,1}^\top \mathbf{x}_{n_q} + \mathbf{r}_{2,1}^\top \mathbf{x}_{n_q} & \cdots & X_{n_q} \mathbf{r}_{1,1}^\top \mathbf{x}_{n_q} + \mathbf{r}_{n_q,1}^\top \mathbf{x}_{n_q} \\ X_1 & X_2 + 1 & \cdots & X_{n_q} + 1 \end{bmatrix} \in \mathbb{R}^{3 \times n_q} \end{aligned} \quad (2.30)$$

Contrary to the fully actuated case, the finger joint torques and displacements are not the controllable inputs of the system, only \dot{q}_a and t_a are. These equations show that for an underactuated finger, the exerted contact wrench and twist depend heavily on the geometry of the chosen underactuation system, on the finger configuration, and on uncontrollable joint torques and velocities: their influence is visible both in \mathbf{t} and \mathbf{T}_f . For example, from Equations (2.29) and (2.30), $\tilde{\xi}_f$ can be expressed as follows:

$$\tilde{\xi}_f = \begin{bmatrix} -X_1 \mathbf{r}_{1,1}^\top \mathbf{y}_{n_q} \dot{q}_a - (X_2 \mathbf{r}_{1,1}^\top \mathbf{y}_{n_q} + \mathbf{r}_{2,1}^\top \mathbf{y}_{n_q}) \dot{q}_2 - \cdots - (X_{n_q} \mathbf{r}_{1,1}^\top \mathbf{y}_{n_q} + \mathbf{r}_{n_q,1}^\top \mathbf{y}_{n_q}) \dot{q}_{n_q} \\ X_1 \mathbf{r}_{1,1}^\top \mathbf{x}_{n_q} \dot{q}_a + (X_2 \mathbf{r}_{1,1}^\top \mathbf{x}_{n_q} + \mathbf{r}_{2,1}^\top \mathbf{x}_{n_q}) \dot{q}_2 + \cdots + (X_{n_q} \mathbf{r}_{1,1}^\top \mathbf{x}_{n_q} + \mathbf{r}_{n_q,1}^\top \mathbf{x}_{n_q}) \dot{q}_{n_q} \\ X_1 \dot{q}_a + (X_2 + 1) \dot{q}_2 + \cdots + (X_{n_q} + 1) \dot{q}_{n_q} \end{bmatrix} \quad (2.31)$$

$$\tilde{\xi}_f = \underbrace{\begin{bmatrix} -X_1 \mathbf{r}_{1,1}^\top \mathbf{y}_{n_q} \\ X_1 \mathbf{r}_{1,1}^\top \mathbf{x}_{n_q} \\ X_1 \end{bmatrix}}_{\tilde{\mathbf{J}}_a \dot{q}_a, \text{ controlled}} \dot{q}_a + \underbrace{\begin{bmatrix} -X_2 \mathbf{r}_{1,1}^\top \mathbf{y}_{n_q} - \mathbf{r}_{2,1}^\top \mathbf{y}_{n_q} & \cdots & -X_{n_q} \mathbf{r}_{1,1}^\top \mathbf{y}_{n_q} - \mathbf{r}_{n_q,1}^\top \mathbf{y}_{n_q} \\ X_2 \mathbf{r}_{1,1}^\top \mathbf{x}_{n_q} + \mathbf{r}_{2,1}^\top \mathbf{x}_{n_q} & \cdots & X_{n_q} \mathbf{r}_{1,1}^\top \mathbf{x}_{n_q} + \mathbf{r}_{n_q,1}^\top \mathbf{x}_{n_q} \\ X_2 + 1 & \cdots & X_{n_q} + 1 \end{bmatrix}}_{\tilde{\mathbf{J}}_j \dot{\mathbf{q}}_j, \text{ not controlled}} \begin{bmatrix} \dot{q}_2 \\ \vdots \\ \dot{q}_{n_q} \end{bmatrix} \quad (2.32)$$

This shows that in the underactuated case, the product $\tilde{\mathbf{J}}_f \mathbf{T}_f$ can be decomposed in two components: $\tilde{\mathbf{J}}_a$ the actuator Jacobian, and $\tilde{\mathbf{J}}_j$ the underactuated joints Jacobian:

$$\tilde{\xi}_f = [\tilde{\mathbf{J}}_a \quad \tilde{\mathbf{J}}_j] \begin{bmatrix} \dot{q}_a \\ \dot{\mathbf{q}}_j \end{bmatrix} \quad (2.33)$$

The same reasoning can be applied for the relation between joint torques and contact wrench:

$$\begin{bmatrix} t_a \\ \mathbf{t}_j \end{bmatrix}^\top = \tilde{\mathbf{w}}_f^\top [\tilde{\mathbf{J}}_a \quad \tilde{\mathbf{J}}_j] \quad (2.34)$$

From these relationships, it is clear that $\tilde{\xi}_f$ and $\tilde{\mathbf{w}}_f$ are not fully controllable: all the contributions from $\dot{\mathbf{q}}_j$ or \mathbf{t}_j can vary independently from the one depending on \dot{q}_a or t_a , the only controllable variables. It also shows that the contact positions are not fully controllable either. With regard to this, underactuated grippers can be considered more like gripper with deformable jaws than as classic fully actuated multifingered gripper [7].

For an underactuated gripper with n_{qg} degrees of freedom, the previous relationships also hold. Let $\dot{\mathbf{q}}_{ag}$ be the concatenation of the actuator velocities of the whole gripper,

and $\dot{\mathbf{q}}_{jg}$ be the concatenation of the velocities of the joints not directly linked to an actuator across the whole gripper. $\boldsymbol{\omega}_{ag}$ is defined as:

$$\boldsymbol{\omega}_{ag} = \begin{bmatrix} \dot{\mathbf{q}}_{ag} \\ \dot{\mathbf{q}}_{jg} \end{bmatrix} \in \mathbb{R}^{n_{ag}} \quad (2.35)$$

Likewise, let \mathbf{t}_{ag} be the concatenation of the actuator torques of the whole gripper, and \mathbf{t}_{jg} be the concatenation of the torques at the joints not directly linked to an actuator across the whole gripper. \mathbf{t}_g is defined as:

$$\mathbf{t}_g = \begin{bmatrix} \mathbf{t}_{ag} \\ \mathbf{t}_{jg} \end{bmatrix} \in \mathbb{R}^{n_{ag}} \quad (2.36)$$

\mathbf{T}_g and $\tilde{\mathbf{J}}_g$ can be created by aggregating respectively the transmission matrices and Jacobian matrices corresponding to each fingers, and by reorganizing their rows and columns to match the joints ordering of $\boldsymbol{\omega}_{ag}$ and \mathbf{t}_g . The two matrix $\tilde{\mathbf{J}}_{ag}$ (gripper actuators Jacobian) and $\tilde{\mathbf{J}}_{jg}$ (gripper underactuated joints Jacobian) can then be defined as in the single finger case.

The matrices \mathbf{J}_a , \mathbf{J}_j , \mathbf{J}_{ag} and \mathbf{J}_{jg} can be constructed with \mathbf{H}_g similarly to the fully-actuated case, to take into account the effect of the contact models on transmitted wrenches and twists.

2.1.5 Grasp Map

To completely describe a grasp, the effect of the contact wrenches and twists on the object needs to be assessed. Each contact wrench or twist is known at the contact point, and is expressed in its respective contact frame C_i , as depicted in Figure 2.6. Thus, it is required to compute the force and twist exerted by each contact point at the object frame origin, and express it in this frame or any given reference frame (i.e. world frame).

First, let \mathbf{w}_o be the contact wrench applied at the object frame origin expressed in the object frame O , and \mathbf{w}_{oc_i} be the contact wrench applied at the object frame origin expressed in the contact frame C_i , with $\mathbf{d}_{oc_i} = [d_x, d_y, d_z]^\top$ the vector from the object frame origin to the contact frame origin, and \mathbf{R}_{oc_i} the rotation between the object frame O and the contact frame C_i . By changing the application point of the contact wrench from the contact point to the object frame origin, one obtains:

$$\mathbf{w}_{oc_i} = \begin{bmatrix} \mathbf{I}_3 & 0 \\ \widehat{\mathbf{d}}_{oc_i} & \mathbf{I}_3 \end{bmatrix} \tilde{\mathbf{w}}_i \quad (2.37)$$

with $\widehat{\mathbf{d}}_{oc_i}$ the cross product matrix associated with \mathbf{d}_{oc_i} :

$$\widehat{\mathbf{d}}_{oc_i} = \begin{bmatrix} 0 & -d_z & d_y \\ d_z & 0 & -d_x \\ -d_y & d_x & 0 \end{bmatrix} \quad (2.38)$$

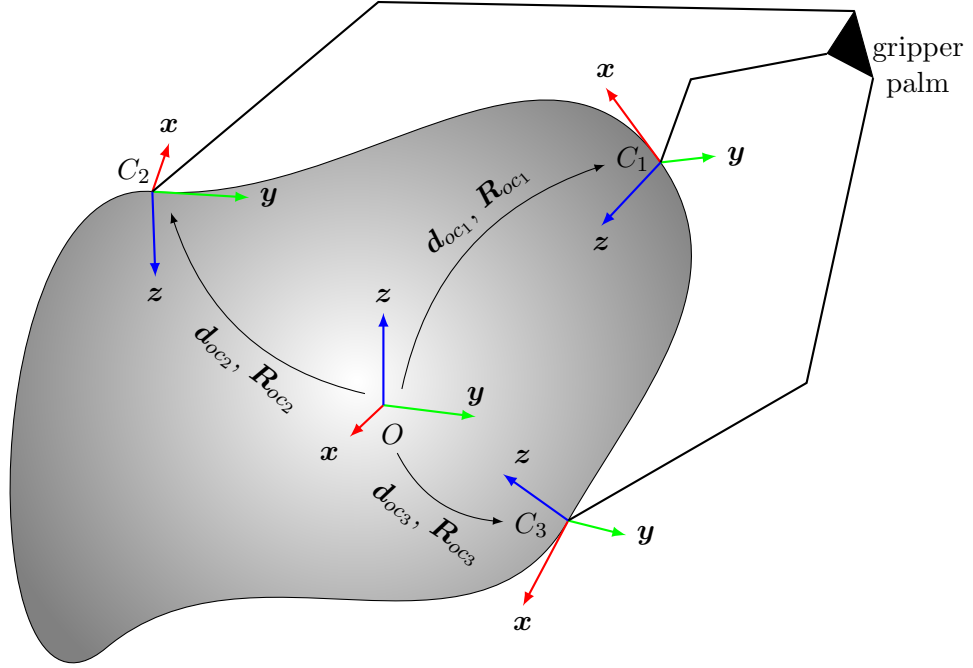


Figure 2.6 – Scheme of three contacts on an object, produced by a three fingered gripper. O is the frame associated with the object, for example aligned on its center of mass and principle inertial axes. C_i is the frame associated with the i^{th} contact point: its origin on the contact point, and with the z axis along the contact normal and toward the object.

then, by expressing \mathbf{w}_{oc_i} and $\tilde{\mathbf{w}}_i$ in the object frame in Equation 2.37, one obtains:

$$\mathbf{w}_o = \begin{bmatrix} \mathbf{I}_3 & 0 \\ \hat{\mathbf{d}}_{oc_i} & \mathbf{I}_3 \end{bmatrix} \begin{bmatrix} \mathbf{R}_{oc_i} & 0 \\ 0 & \mathbf{R}_{oc_i} \end{bmatrix} \tilde{\mathbf{w}}_i \quad (2.39)$$

$$\mathbf{w}_o = \begin{bmatrix} \mathbf{R}_{oc_i} & 0 \\ \hat{\mathbf{d}}_{oc_i} \mathbf{R}_{oc_i} & \mathbf{R}_{oc_i} \end{bmatrix} \tilde{\mathbf{w}}_i \quad (2.40)$$

The contact map is defined as follows:

$$\tilde{\mathbf{G}}_i = \begin{bmatrix} \mathbf{R}_{oc_i} & 0 \\ \hat{\mathbf{d}}_{oc_i} \mathbf{R}_{oc_i} & \mathbf{R}_{oc_i} \end{bmatrix} \in \mathbb{R}^{6 \times 6} \quad (2.41)$$

By reasoning with contact twist $\tilde{\boldsymbol{\xi}}_i$ expressed in contact frame C_i and object twist $\boldsymbol{\xi}_o$ expressed in object frame O , a relation similar to Equation 2.40 exists, and is further detailed in Prattichizzo and Trinkle [37] and Murray et al. [77]. The contact map $\tilde{\mathbf{G}}_i$ allows to relate the contact wrench or twist with the object wrench or twist:

$$\mathbf{w}_o = \tilde{\mathbf{G}}_i \tilde{\mathbf{w}}_i \quad (2.42)$$

$$\tilde{\boldsymbol{\xi}}_i = \tilde{\mathbf{G}}_i^\top \boldsymbol{\xi}_o \quad (2.43)$$

For a gripper creating n_c contacts on the object, the total object wrench is the sum of the individual contributions of each contact wrench, thus \mathbf{w}_o become:

$$\mathbf{w}_o = \tilde{\mathbf{G}}_1 \tilde{\mathbf{w}}_1 + \cdots + \tilde{\mathbf{G}}_{n_c} \tilde{\mathbf{w}}_{n_c} = \left[\tilde{\mathbf{G}}_1 \cdots \tilde{\mathbf{G}}_{n_c} \right] \tilde{\mathbf{w}}_g \quad (2.44)$$

The complete grasp map $\tilde{\mathbf{G}}$ is defined as follows:

$$\tilde{\mathbf{G}} = \left[\tilde{\mathbf{G}}_1 \cdots \tilde{\mathbf{G}}_{n_c} \right] \in \mathbb{R}^{6 \times 6n_c} \quad (2.45)$$

As for the contact map, a relation similar to Equation 2.44 holds between contact twists $\tilde{\boldsymbol{\xi}}_g$ and object twist $\boldsymbol{\xi}_o$. The grasp map gives the relation between all the wrenches and twists produced by the gripper at the contact points, and the total object wrench and twist:

$$\mathbf{w}_o = \tilde{\mathbf{G}} \tilde{\mathbf{w}}_g \quad (2.46)$$

$$\tilde{\boldsymbol{\xi}}_g = \tilde{\mathbf{G}}^\top \boldsymbol{\xi}_o \quad (2.47)$$

In order to take into account the contact model and the constraints on transmissible contact wrench and twist component, the contact map can be redefined as follows, using Equation 2.4:

$$\mathbf{G}_i = \tilde{\mathbf{G}}_i \mathbf{H}_i^\top \in \mathbb{R}^{6 \times n_{\lambda i}} \quad (2.48)$$

The grasp map can also be redefined this way, with n_λ the total number of transmitted wrench or twist components:

$$\mathbf{G} = \tilde{\mathbf{G}} \mathbf{H}_g^\top \in \mathbb{R}^{6 \times n_\lambda} \quad (2.49)$$

When used in Equations (2.46) and (2.47), this redefinition of the grasp map allows to relate object wrench \mathbf{w}_o and twist $\boldsymbol{\xi}_o$ to transmitted contact wrenches \mathbf{w}_g and twists $\boldsymbol{\xi}_g$ respectively. The rows of \mathbf{G} correspond to the transmitted contact wrench component contributions to each object wrench components. Each row of \mathbf{G}^\top represents the object twist component contributions to each transmitted contact twist component. The relations existing between the different quantities and matrices presented previously are summarized on Figure 2.7.

The grasp map \mathbf{G} and the gripper Jacobian \mathbf{J}_g (and for an underactuated gripper, the actuator Jacobian matrix \mathbf{J}_{ag} and underactuated joints Jacobian matrix \mathbf{J}_{jg}) are sufficient to fully describe the grasping of an object by a gripper with a fixed contact grasp, provided that the contact wrenches enforce the friction constraints. For a fully actuated gripper, the fundamental grasping constraint [37, 77] is obtained by equating Equations (2.20) and (2.47):

$$\mathbf{J}_g \dot{\mathbf{q}}_g = \mathbf{G}^\top \boldsymbol{\xi}_o \quad (2.50)$$

which becomes for an underactuated gripper:

$$\begin{bmatrix} \mathbf{J}_{ag} & \mathbf{J}_{jg} \end{bmatrix} \begin{bmatrix} \dot{\mathbf{q}}_{ag} \\ \dot{\mathbf{q}}_{jg} \end{bmatrix} = \mathbf{G}^\top \boldsymbol{\xi}_o \quad (2.51)$$

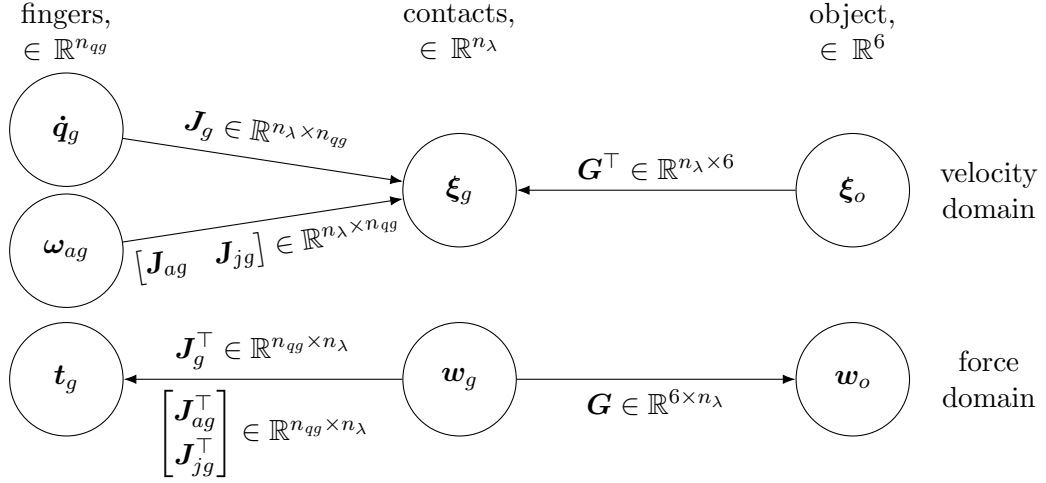


Figure 2.7 – Diagram of the existing relations between quantities linked to the fingers, the contacts and the object.

This relation allows to relate joint velocities $\dot{\mathbf{q}}_g$ (or $\boldsymbol{\omega}_{ag} = [\dot{\mathbf{q}}_{ag} \quad \dot{\mathbf{q}}_{jg}]^\top$ for an underactuated gripper) to the object twist $\boldsymbol{\xi}_o$. It relies on the fact that for a fixed contact (that is a contact that cannot slide in directions specified by the matrix \mathbf{H}_g), the contact twists seen by the phalanges are the same as the contact twists seen by the object.

2.2 Grasp Characterisation

The required formalism for grasp modeling has been given in the previous section. From the grasp description, grasps can be classified, and desirable properties can be extracted. In the following these properties are presented, together with different grasp quality metrics. Finally, the rationale behind the grasp quality metric chosen in this work is presented.

2.2.1 Grasp Classifications & Desirable Properties

Grasp categories having specific features can be determined by studying algebraic properties of the grasp map \mathbf{G} and Jacobian \mathbf{J}_g , as well as their transpose.

Grasp Classifications

To qualify a grasp, one of the main information is which set of twists or wrenches the gripper can apply to the object, and conversely in which conditions the gripper can resist any disturbing wrench and prevent any unwanted object motions.

Four main grasp categories can be identified from the kernel of the four matrices \mathbf{G} , \mathbf{G}^\top , \mathbf{J}_g and \mathbf{J}_g^\top . These categories are summarized in Table 2.4 and are further described

in the following. For an in depth mathematical development of these categories, see Prattichizzo and Trinkle [37].

A grasp is considered *graspable* if $\ker(\mathbf{G})$ is non-trivial (different from the null vector). Such grasp admits a set of contact wrenches \mathbf{w}_g that do not influence the total object wrench \mathbf{w}_o . These contact wrenches are called *internal object forces*. They are the key factor of the grasp tightness, which make them essential for grasp relying on friction. Indeed, the ability to change contact wrenches without modifying object wrench allows to enforce friction constraints more easily.

An *indeterminate* grasp has a non-trivial $\ker(\mathbf{G}^\top)$. For these grasps, there are object twists $\boldsymbol{\xi}_o$ that describe object motions, but without any contact twists in their respective constrained directions $\boldsymbol{\xi}_g$. Thus, it is impossible to control or prevent such object motion with any contact point motions. These object twists are called *internal object twists*. In general, the existence of such twists are not desirable when the goal is to manipulate the object, or if the grasp needs to maintain the object still.

When $\ker(\mathbf{J}_g)$ is non-trivial, the grasp is said to be *redundant*. In such grasp, there are gripper joint velocities $\dot{\mathbf{q}}_g$ that do not create contact twists $\boldsymbol{\xi}_g$. Thus, there is a set of finger motions which is independent from object motions. These joint velocities are called *internal hand velocities*. This property can be used in a similar way to the redundancy in robotic manipulator arm: the additional degrees of freedom can be used to reposition fingers during the grasp, for example to avoid collisions with the environment, or to keep joints as far as possible from their mechanical limits or from any singularity.

If $\ker(\mathbf{J}_g^\top)$ is non-trivial, the grasp is called a *defective* grasp. This category can have contact wrenches \mathbf{w}_g that are not related to gripper joint torques \mathbf{t}_g . These wrenches are called *internal hand forces*: they cannot be produced through the gripper joint torques, however the gripper mechanical structure itself can withstand them. It is preferable to avoid this case, as a gripper involved in a defective grasp cannot exert arbitrary contact wrenches, diminishing its ability to successfully manipulate the object or maintain friction constraints.

Desirable Properties

A grasp has two main desirable properties [37, 86]:

- The ability to resist external disturbances in any direction, that is the ability for the grasp to withstand any object wrenches or twists. This is done by ensuring object immobility thanks to the contact position or the friction forces. When the immobility is ensured by friction forces, the gripper also needs to be able to control internal forces.
- The ability for the gripper in the given grasp configuration to transmit any motion to the object. This requires to be able to control all possible object twists and wrenches. This property depends both on the contacts between the object and the gripper and on the gripper configuration itself. This property is of particular interest when the goal is to perform dexterous manipulation of the object, but

condition on matrix kernel	grasp category	many-to-one relationship	produced internal twist or wrench
$\ker(\mathbf{G}) \neq \mathbf{0}$	Graspable	$\xi_g \rightarrow \xi_o$ $w_g \rightarrow w_o$	internal object forces
$\ker(\mathbf{G}^\top) \neq \mathbf{0}$	Indeterminate	$\xi_o \rightarrow \xi_g$ $w_o \rightarrow w_g$	internal object twists
$\begin{cases} \ker(\mathbf{J}_g) \neq \mathbf{0} \\ \text{or} \\ \ker(\mathbf{J}_{ag}) \neq \mathbf{0} \end{cases}$	redundant	$\begin{cases} \dot{q}_g \rightarrow \xi_g \\ t_g \rightarrow w_g \\ \text{or} \\ \dot{q}_{ag} \rightarrow \xi_g \\ t_{ag} \rightarrow w_g \end{cases}$	internal hand velocities
$\begin{cases} \ker(\mathbf{J}_g^\top) \neq \mathbf{0} \\ \text{or} \\ \ker(\mathbf{J}_{ag}^\top) \neq \mathbf{0} \end{cases}$	Defective	$\begin{cases} \xi_g \rightarrow \dot{q}_g \\ w_g \rightarrow t_g \\ \text{or} \\ \xi_g \rightarrow \dot{q}_{ag} \\ w_g \rightarrow t_{ag} \end{cases}$	internal hand forces

Table 2.4 – Summary of main grasp categories [37].

less pertinent when considering a grasping task that does not require fine object manipulation.

Regarding the ability to maintain the object still for any external disruptive wrench, the grasp needs to be able to enforce any object wrench, in order to resist this disruptive

wrench. For that, the grasp should not be indeterminate. Indeed, an indeterminate grasp admits object wrenches that are independent from contact wrenches. Thus, a necessary condition for a grasp to be able to resist any arbitrary object wrench is $\ker(\mathbf{G}^\top) = \mathbf{0}$, or also $\dim(\text{Im}(\mathbf{G})) = 6$. Any 3D grasp formed with three non-colinear point contacts with friction, or two soft-finger contacts satisfies this minimum condition that needs to be fulfilled [37]. Cases with more contacts are preferable in practice.

Once this first necessary condition is met, the object immobility can then be obtained through two mechanisms: form-closure or force-closure.

- Form-closure is a purely geometric consideration: a grasp is form-closure if the contact positions on the object surface make it strictly impossible to move the object. An in depth mathematical analysis of this property is available in Prattichizzo and Trinkle [37].
- Force-closure is a static or dynamic consideration. a grasp is force-closure if for any external wrench $\mathbf{w}_e \in \mathbb{R}^6$, there are controllable contact wrenches \mathbf{w}_g belonging to the friction cone FC (defined by the constraints described in subsection 2.1.1) such as [37, 77]:

$$\mathbf{G}\mathbf{w}_g = -\mathbf{w}_e, \quad |\mathbf{w}_g \in FC \quad (2.52)$$

A necessary condition for force-closure is that the grasp has internal object forces, that is it needs to belong to the graspable category ($\ker(\mathbf{G}) \neq \mathbf{0}$). Indeed, the existence of internal object forces are required to enforce the friction constraints. Moreover, internal object forces need to be inside the friction cone, that is:

$$\exists \mathbf{w}_g \in \ker(\mathbf{G}) \mid \mathbf{w}_g \in FC \quad (2.53)$$

A grasp that verify this condition and for which $\ker(\mathbf{G}^\top) = \mathbf{0}$ and $\ker(\mathbf{G}) \neq \mathbf{0}$ has frictional form-closure [37].

The last necessary condition for force-closure is to be able to effectively apply the contact wrenches required to withstand the external disruptive wrench \mathbf{w}_e . For that, the grasp needs to control all internal object forces. This is true if and only if:

$$\ker(\mathbf{G}) \cap \ker(\mathbf{J}_g^\top) = \mathbf{0} \quad (2.54)$$

A frictional form-closure grasp that verify this condition is force-closure [37]. The internal object forces are in $\ker(\mathbf{G})$, and the internal hand forces are in $\ker(\mathbf{J}_g^\top)$. It means that a grasp can be both defective and force-closure: the condition simply states that internal object forces and internal hand forces must be two non-overlapping wrench sets.

In the general case, assessing if a grasp is force-closure is a complex task mainly because of the friction cone constraints, which are quadratic, and require to use non-linear programming techniques to be solved, or require linearization of the friction cone to fall down into Linear Programming Problem.

For an underactuated gripper, the grasp categories and desirable properties presented above still hold. In particular, the criterion on \mathbf{G} and \mathbf{G}^\top null spaces have the same interpretation, whether the grasp is performed by a fully-actuated gripper or an underactuated one. However, in an underactuated gripper, only actuator velocities and torques are controllable. For the underactuated joints, the joint torques and velocities are determined by the characteristics of the chosen underactuation mechanism and its configuration, that is the joint positions. Thus, in this case the interpretation of the null space of \mathbf{J}_g and \mathbf{J}_g^\top can be misleading.

Indeed, among the contact twists or wrenches involving some underactuated joints, some of them are not achievable through actuator commands alone. They can be produced passively by the underactuated system, but in this case it will be endured and not controlled. Such situation can be interpreted as a defective grasp. Likewise, if the grasp is redundant, but this redundancy is only associated to underactuated joints, the property loses its interest: it cannot be used to purposely reconfigure the gripper. **To take this into account, the defectiveness and redundancy properties can be evaluated for an underactuated gripper on \mathbf{J}_{ag} and \mathbf{J}_{ag}^\top , the grasp actuator Jacobian, as introduced in subsection 2.1.4.** The grasp categories are summarized in Table 2.4 for both fully-actuated and underactuated grippers. The differences between the two cases are highlighted in the following example.

Example: Effect of the Underactuation on the Grasp Properties

To highlight the specificities of the underactuation, and its effect on grasp properties, the null space of the Grasp Map and gripper Jacobian will be computed on a simple case study, firstly with a fully-actuated gripper, and then with an underactuated one.

Fully-actuated Gripper

The considered gripper has two fingers and six revolute joints, three on each finger. It is displayed when grasping a sphere in Figure 2.8. In the current configuration, both fingers lie in the figure plane. Joints q_1^1 and q_1^2 have their rotation axis along a vertical direction, allowing an out of plane motion. The other joints have their axis of rotation perpendicular to the figure plane. Thus, the grasp analysis is made in 3D. It is assumed that the contact C_1 is modeled with a point contact with friction, and that contact C_2 is modeled with a soft-finger contact (for example due to a difference of material of the phalanx surfaces).

First, lets compute the Grasp Map \mathbf{G} :

$$\mathbf{R}_{oc_1} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad \mathbf{R}_{oc_2} = \begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \end{bmatrix} \quad (2.55)$$

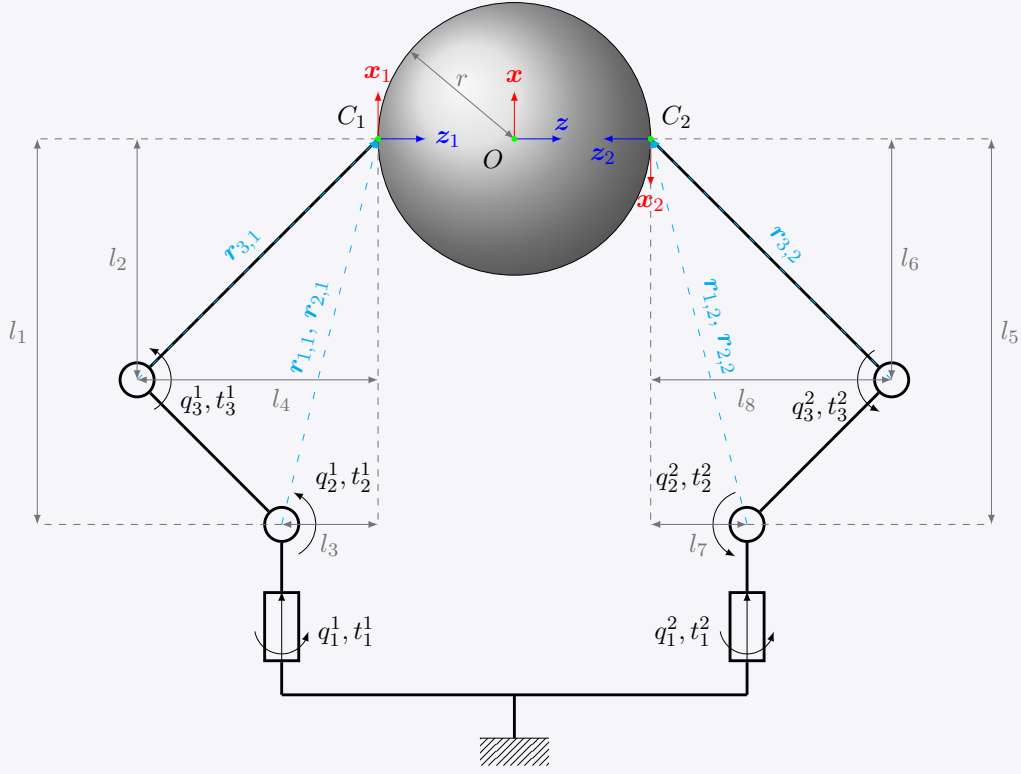


Figure 2.8 – A sphere grasped by a fully-actuated two fingered gripper with six joints.

$$\mathbf{d}_{oc_1} = [0, 0, -r]^\top \quad \mathbf{d}_{oc_2} = [0, 0, r]^\top \quad (2.56)$$

$$\hat{\mathbf{d}}_{oc_1} = \begin{bmatrix} 0 & r & 0 \\ -r & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad \hat{\mathbf{d}}_{oc_2} = \begin{bmatrix} 0 & -r & 0 \\ r & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad (2.57)$$

$$\hat{\mathbf{d}}_{oc_1} \mathbf{R}_{oc_1} = \begin{bmatrix} 0 & r & 0 \\ -r & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad \hat{\mathbf{d}}_{oc_2} \mathbf{R}_{oc_2} = \begin{bmatrix} 0 & -r & 0 \\ -r & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad (2.58)$$

Then, by combining Equations (2.55) and (2.58), the contact maps can be computed:

$$\tilde{\mathbf{G}}_{c_1} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & r & 0 & 1 & 0 & 0 \\ -r & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \quad \tilde{\mathbf{G}}_{c_2} = \begin{bmatrix} -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & -r & 0 & -1 & 0 & 0 \\ -r & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 \end{bmatrix} \quad (2.59)$$

The \mathbf{H} matrices corresponding to the contacts are:

$$\mathbf{H}_{c_1} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix} \quad \mathbf{H}_{c_2} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \quad (2.60)$$

Thus, the Grasp Map is:

$$\mathbf{G} = [\tilde{\mathbf{G}}_{c_1} \mathbf{H}_{c_1}^\top \quad \tilde{\mathbf{G}}_{c_2} \mathbf{H}_{c_2}^\top] \quad (2.61)$$

$$\mathbf{G} = \begin{bmatrix} 1 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & -1 & 0 \\ 0 & r & 0 & 0 & -r & 0 & 0 \\ -r & 0 & 0 & -r & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 \end{bmatrix} \quad (2.62)$$

Here, $\ker(\mathbf{G}) \neq \mathbf{0}$ and $\dim(\ker(\mathbf{G})) = 1$, the grasp belong to the *graspable* category: a contact wrench in $\ker(\mathbf{G})$ is for example $[0, 0, 1, 0, 0, 1, 0]^\top$. This direction corresponds to the segment connecting the two contact points: the internal object forces are along this vector. Moreover, $\ker(\mathbf{G}^\top) = \mathbf{0}$, which mean that the grasp is not *indeterminate*: every object wrench can be related to some contact wrenches. Both properties are necessary conditions for force-closure, and can be useful to maintain the object still.

To compute the gripper Jacobian \mathbf{J}_g , the vectors between the k^{th} joint frame and the corresponding contact frame ($\mathbf{r}_{k,i}$ Figure 2.8) need to be expressed in contact frame. Generally, all their components are function of the configuration of the gripper joints. Here, they are computed in the specific configuration where all joints lie on the same plane:

$$\mathbf{r}_{1,1} = \mathbf{r}_{2,1} = [l_1, 0, l_3]^\top \quad (2.63)$$

$$\mathbf{r}_{1,2} = \mathbf{r}_{2,2} = [-l_5, 0, l_7]^\top \quad (2.64)$$

$$\mathbf{r}_{3,1} = [l_2, 0, l_4]^\top \quad (2.65)$$

$$\mathbf{r}_{3,2} = [-l_6, 0, l_8]^\top \quad (2.66)$$

Then, the Jacobian of the two fingers can be computed:

$$\tilde{\mathbf{J}}_1 = \begin{bmatrix} 0 & l_3 & l_4 \\ -l_3 & 0 & 0 \\ 0 & -l_1 & -l_2 \\ 1 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & 0 \end{bmatrix} \quad \tilde{\mathbf{J}}_2 = \begin{bmatrix} 0 & l_7 & l_8 \\ l_7 & 0 & 0 \\ 0 & l_5 & l_6 \\ -1 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & 0 \end{bmatrix} \quad (2.67)$$

Finally, the gripper Jacobian is:

$$\mathbf{J}_g = \begin{bmatrix} \mathbf{H}_{c1} \tilde{\mathbf{J}}_{f1} & \mathbf{0} \\ \mathbf{0} & \mathbf{H}_{c2} \tilde{\mathbf{J}}_{f2} \end{bmatrix} \quad (2.68)$$

$$\mathbf{J}_g = \begin{bmatrix} 0 & l_3 & l_4 & 0 & 0 & 0 \\ -l_3 & 0 & 0 & 0 & 0 & 0 \\ 0 & -l_1 & -l_2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & l_7 & l_8 \\ 0 & 0 & 0 & l_7 & 0 & 0 \\ 0 & 0 & 0 & 0 & l_5 & l_6 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad (2.69)$$

Here, $\ker(\mathbf{J}_g) = \mathbf{0}$: the grasp is not *redundant*. There is no combination of joint motions that are independent from contact motions.

For this gripper and grasp configuration, $\ker(\mathbf{J}^\top) \neq \mathbf{0}$ and $\dim(\ker(\mathbf{J}^\top)) = 1$: the grasp is *defective*. A contact wrench belonging to the null space of \mathbf{J}^\top , that is an internal hand force, is for example $[0, 0, 0, 0, 0, 0, 1]^\top$: the gripper is unable to exert a torque in the normal direction of the contact C_2 . All the other contact wrenches can be generated by the gripper, and used to produce an object wrench able to cancel external disturbances, hence preventing any movement of the object. Moreover, an external object wrench generating a contact wrench belonging to the internal hand forces can still be absorbed by the gripper structure, which also prevents object motion. However, both mechanisms are possible only if the contact wrenches stay inside the friction cone. For that, the internal object forces, that are used to tighten the grasp, needs to be inside it. For this grasp, this condition is met, as the internal object forces are along the normal vector of both contacts. Finally, the gripper needs to be able to exert contact wrenches in the internal object forces. Here, this is the case, as $\ker(\mathbf{G}) \cap \ker(\mathbf{J}_g^\top) = \mathbf{0}$. Thus, this grasp is force-closure.

Underactuated Gripper

The considered underactuated gripper has the same architecture than the fully-actuated one: two identical fingers with three joints each. Joints q_1^1 and q_1^2 are fully-actuated. Joints q_2^1 and q_3^1 (and respectively joints q_2^2 and q_3^2) are driven through an underactuation mechanism.

These underactuation mechanisms are defined by the following transmission matrices and underactuation torques:

$$\mathbf{T}_1 = \begin{bmatrix} X_2^1 & X_3^1 \\ 0 & 1 \end{bmatrix} \quad \mathbf{T}_2 = \begin{bmatrix} X_2^2 & X_3^2 \\ 0 & 1 \end{bmatrix} \quad (2.70)$$

$$\begin{bmatrix} q_2^1 \\ q_1^1 \\ q_3^1 \end{bmatrix} = \mathbf{T}_1 \begin{bmatrix} q_a^1 \\ q_1^1 \\ q_3^1 \end{bmatrix} \quad (2.71)$$

$$t_3^1 = -K_1 \Delta q_3^1 \quad (2.72)$$

$$\begin{bmatrix} q_2^2 \\ q_3^2 \end{bmatrix} = \mathbf{T}_2 \begin{bmatrix} q_a^2 \\ q_3^2 \end{bmatrix}$$

$$t_3^2 = -K_2 \Delta q_3^2$$

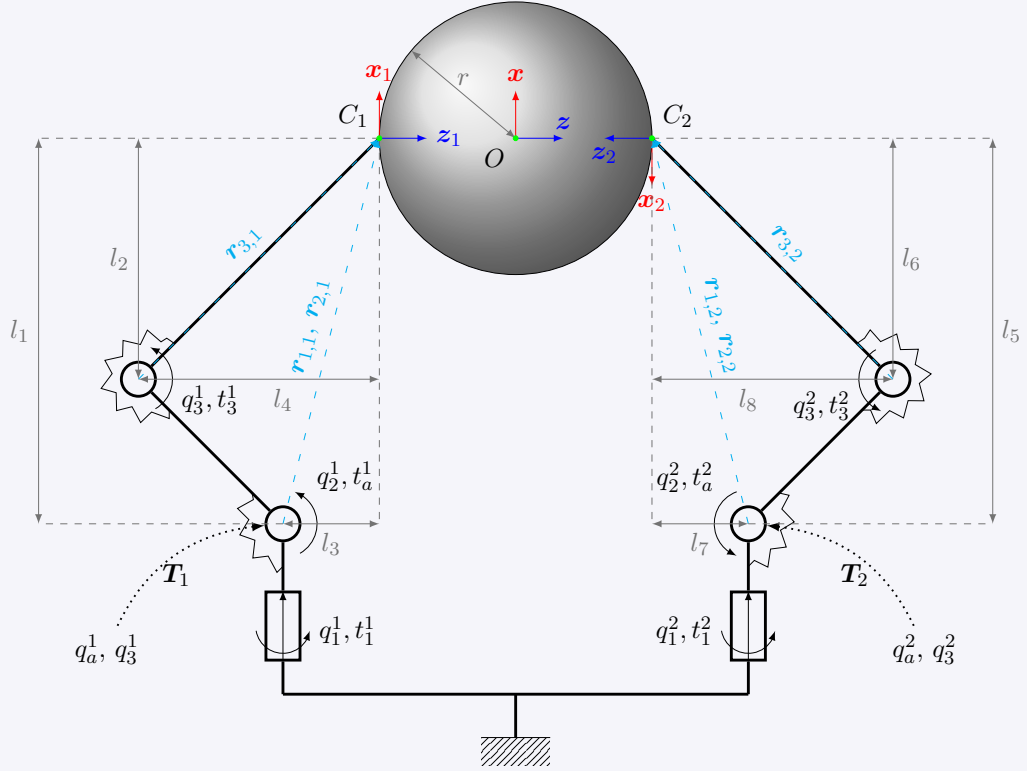


Figure 2.9 – A sphere grasped by an underactuated two fingered gripper with six joints. the underactuation mechanism acts on the two last joints of each fingers.

The X_i^k coefficients depend on the underactuation mechanism characteristics and on the joint configuration. Their analytic formulation and computation is detailed in Birglen et al. [7]. K_1 and K_2 are the stiffness coefficients of the spring of the underactuation system of the finger 1 and 2 respectively. Δq_3^1 and Δq_3^2 are the distance from the rest position of the respective joints.

The gripper is displayed when grasping a sphere in Figure 2.9, in the same configuration as in the fully-actuated case. The same assumptions are made on the type of contacts: point contact with friction for C_1 , soft-finger for C_2 .

in this case, the Grasp Map is identical to the fully-actuated case (Equation 2.62): the grasp is *graspable* ($\ker(\mathbf{G}) \neq \mathbf{0}$), and not *indeterminate* ($\ker(\mathbf{G}^\top) = \mathbf{0}$). The internal object forces are along the same vector as in the fully-actuated case: $[0, 0, 1, 0, 0, 1, 0]^\top$. Thus, this grasp also fulfils the necessary conditions for force-closure.

The difference with the fully-actuated case comes from the actuator Jacobian matrix \mathbf{J}_{ag} and its associated null spaces. To be computed, it first requires to express the gripper transmission matrix \mathbf{T}_g . It can be expressed from Equation 2.70 as follows:

$$\mathbf{T}_g = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & X_2^1 & 0 & 0 & X_3^1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & X_2^2 & 0 & X_3^2 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \quad (2.73)$$

with:

$$[q_1^1, q_2^1, q_1^2, q_2^2, q_3^1, q_3^2]^\top = \mathbf{T}_g [q_1^1, q_a^1, q_1^2, q_a^2, q_3^1, q_3^2]^\top \quad (2.74)$$

This specific joint ordering allows to separate the actuated joints from the joints that are not directly linked to an actuator in two different vectors as follows:

$$\mathbf{q}_{ag} = [q_1^1, q_a^1, q_1^2, q_a^2]^\top \quad \mathbf{q}_{jg} = [q_3^1, q_3^2]^\top \quad (2.75)$$

which gives:

$$\mathbf{q}_g = \mathbf{T}_g \begin{bmatrix} \mathbf{q}_{ag} \\ \mathbf{q}_{jg} \end{bmatrix} \quad (2.76)$$

The gripper Jacobian \mathbf{J}_g can be computed the same way as in the fully-actuated case. It corresponds to the matrix given in Equation 2.69, with the columns and rows ordering adjusted to match the joint ordering of \mathbf{q}_{ag} and \mathbf{q}_{jg} :

$$\mathbf{J}_g = \begin{bmatrix} 0 & l_3 & 0 & 0 & l_4 & 0 \\ -l_3 & 0 & 0 & 0 & 0 & 0 \\ 0 & -l_1 & 0 & 0 & -l_2 & 0 \\ 0 & 0 & 0 & l_7 & 0 & l_8 \\ 0 & 0 & l_7 & 0 & 0 & 0 \\ 0 & 0 & 0 & l_5 & 0 & l_6 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad (2.77)$$

Then, the gripper actuator Jacobian matrix \mathbf{J}_{ag} can be extracted from the product $\mathbf{J}_g \mathbf{T}_g$, by selecting the first four columns, corresponding to the four actuated joints:

$$\mathbf{J}_g \mathbf{T}_g = \begin{bmatrix} 0 & X_2^1 l_3 & 0 & 0 & X_3^1 l_3 + l_4 & 0 \\ -l_3 & 0 & 0 & 0 & 0 & 0 \\ 0 & -X_2^1 l_1 & 0 & 0 & -X_3^1 l_1 - l_2 & 0 \\ 0 & 0 & 0 & X_2^2 l_7 & 0 & X_3^2 l_7 + l_8 \\ 0 & 0 & l_7 & 0 & 0 & 0 \\ 0 & 0 & 0 & X_2^2 l_5 & 0 & X_3^2 l_5 + l_6 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad (2.78)$$

$$= [\mathbf{J}_{ag} \quad \mathbf{J}_{jg}] \quad (2.79)$$

$$\mathbf{J}_{ag} = \begin{bmatrix} 0 & X_2^1 l_3 & 0 & 0 \\ -l_3 & 0 & 0 & 0 \\ 0 & -X_2^1 l_1 & 0 & 0 \\ 0 & 0 & 0 & X_2^2 l_7 \\ 0 & 0 & l_7 & 0 \\ 0 & 0 & 0 & X_2^2 l_5 \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad (2.80)$$

The grasp is not redundant, as $\ker(\mathbf{J}_{ag}) = \mathbf{0}$. This is expected, as the fully-actuated version is not redundant either, and the underactuation mechanism reduces the number of controllable joints.

The grasp is defective, as in the fully actuated case, $\ker(\mathbf{J}_{ag}^\top) \neq \mathbf{0}$. Here, the contact wrench $[0, 0, 0, 0, 0, 0, 1]^\top$ still belong to the null space of \mathbf{J}_{ag}^\top : the gripper is unable to exert a torque in the normal direction of C_2 . However, in the underactuated case, the null space is of higher dimension: $\dim(\ker(\mathbf{J}_{ag}^\top)) = 3$. Two other contact wrenches belonging to the kernel of \mathbf{J}_{ag}^\top (and that were not in the kernel of \mathbf{J}_g^\top) are for example:

$$\mathbf{v}_1 = \left[\frac{1}{l_3}, 0, \frac{1}{l_1}, 0, 0, 0, 0 \right]^\top \quad \mathbf{v}_2 = \left[0, 0, 0, \frac{1}{l_7}, 0, -\frac{1}{l_5}, 0 \right]^\top \quad (2.81)$$

These contact wrenches can be produced only with the contribution of t_3^1 and t_3^2 , and correspond to wrenches along $\mathbf{r}_{1,1}$ and $\mathbf{r}_{1,2}$ respectively. In the underactuated case, the torques on these joints depend on the underactuation mechanism, are fixed for a given joint configuration (see Equation 2.72), and thus are not controllable. An external contact wrench with a component in one of these directions will lead to a motion of the underactuated joints, until the joints not directly linked to an actuator (here q_3^1 and q_3^2) reach a configuration where they produce a torque that balance the contact wrench.

Thus, the underactuation mechanism generates new dimensions in the defective space compared to the fully-actuated equivalent.

Regarding the force-closure property, the internal object forces are along the normal vector of both contacts, as for the fully-actuated gripper, thus they are inside the friction cone. Moreover, $\ker(\mathbf{G}) \cap \ker(\mathbf{J}_{ag}^\top) = \mathbf{0}$: the gripper is able to exert contact wrenches inside the internal object forces, the grasp is force-closure.

However, the new defective contact wrenches (\mathbf{v}_1 and \mathbf{v}_2) depend on l_1 , l_3 , l_5 and l_7 , quantities that depend on the gripper joint configuration. In particular, for the configuration where $l_1 = l_5 = 0$, $\ker(\mathbf{G}) \cap \ker(\mathbf{J}_{ag}^\top) \neq \mathbf{0}$: the grasp loses the force-closure property.

Due to the uncontrollable contribution of t_3^1 and t_3^2 to the contact wrench, it is not possible to produce any arbitrary internal object forces for every given gripper

configuration. The contact wrench \mathbf{w}_g produced by a given torque vector \mathbf{t}_g is:

$$\mathbf{w}_g = \begin{bmatrix} \mathbf{J}_{ag}^\top \\ \mathbf{J}_{jg}^\top \end{bmatrix}^+ \mathbf{t}_g \quad (2.82)$$

with \mathbf{M}^+ the pseudo-inverse of a matrix \mathbf{M} . Solving this system for $\mathbf{w}_g \in \ker(\mathbf{G})$ allows to find the constraints on joint torques and joint angles that need to be enforced to produce a contact wrench inside the internal object forces.

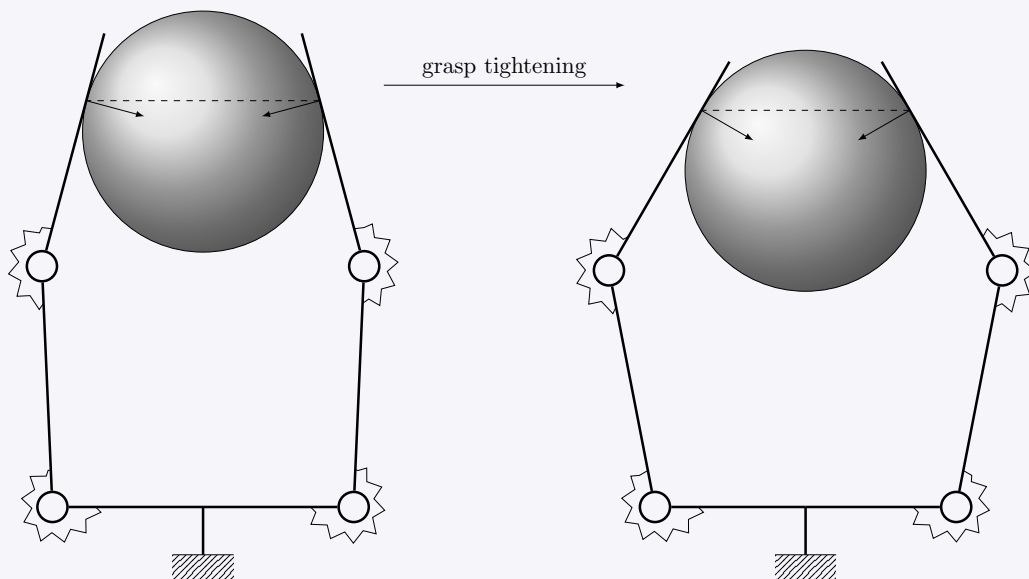


Figure 2.10 – Example of an underactuated gripper with two two-phalanx fingers grasping a sphere. On the left a grasp configuration at equilibrium with a small tightening effort, on the right the grasp configuration at equilibrium after increasing the tightening force. The arrows show the contact normals, the dashed lines show the orientation of the internal object forces. Tightening the grasp changes the orientation of the friction cone relative to the internal object forces.

As long as these conditions are not met, tightening the grasp produces a contact wrench that has a component outside of the internal object forces, which produces an object wrench. This object wrench will produce a motion of the object, through a passive motion of the underactuated joints. This motion stops if q_2^1 , q_3^1 , q_2^2 and q_3^2 reach a configuration where the produced torques enforce the constraints mentioned above. It can also stop when the joint limits are reached, or when a new contact is created with another phalanx or the palm. However, this motion changes the position of the gripper phalanges relative to the object. This can change the contact normal orientations and contact positions through a rolling motion of the object on the phalanges for example. When it happens, it changes

the orientation of the friction cone relative to the internal object forces. This is illustrated in Figure 2.10. A configuration where the internal object forces leave the friction cone can be reached: in this case, the grasp is no longer force-closure.

Thus, the specific behavior of underactuated grippers makes the assessment of the force-closure property much more complex than in the fully-actuated case.

2.2.2 Overview of Existing Grasp Quality Metrics

In the previous section, different desirable grasp properties have been shown. For a given gripper and object, it is very common to have several grasp configurations that verify one of the above property about grasp features. Thus, the different possible grasps have to be ranked according to a quality measure. Some of the main existing metrics are presented in the following, and are summarized in Table 2.5. A more complete list, with more in depth metric descriptions is available in Roa and Suárez [86].

Metrics Depending on Contact Point Positions Only

This group of quality measures takes into account object properties, such as its shape, weight, friction coefficient, and contact positions on it. It can be divided in four sub-categories:

- Metrics based on the Grasp Map,
- Metrics based on geometric relations between contact positions,
- Metrics assessing the frictional form-closure property,
- Metrics taking into account limits on the contact wrench magnitudes

Using the Grasp Map The main principle of these metrics is to assess if the grasp is not indeterminate (that is if $\dim(\text{Im}(\mathbf{G})) = 6$), and then determine to what extent. These metrics rely on the computation of the singular values of \mathbf{G} , that is the positive square root of the eigenvalues of $\mathbf{G}\mathbf{G}^T$ [86]. There are three main metrics in this category: the minimum singular value of \mathbf{G} [87, 88], the volume of the ellipsoid in the wrench space [87], and the grasp isotropy index [88]. For all three metrics, a positive value represents a necessary condition for force-closure [87], as a null value indicates that the grasp is indeterminate.

- The minimum singular value of \mathbf{G} is simply expressed as follows, with σ_i the singular values of \mathbf{G} numbered in descending order [87, 88]:

$$Q_{MSV} = \sigma_6 \quad (2.83)$$

It gives an information on a worst case scenario: Q_{MSV} indicates how far the grasp is from losing the ability to resist wrench in the most unfavorable direction. Indeed, when $Q_{MSV} = 0$, the grasp is unable to withstand a disruptive wrench in one direction at least. A larger Q_{MSV} corresponds to a smaller maximum transmission ratio between \mathbf{w}_o and \mathbf{w}_f , thus minimizing the maximum contact load required for a given object wrench [88].

- The volume of the ellipsoid in the wrench space is [87]:

$$Q_{VEW} = \sqrt{\det(\mathbf{G}\mathbf{G}^\top)} = \prod_{n=1}^6 \sigma_n \quad (2.84)$$

A larger Q_{VEW} value means that the ellipsoid of admissible object wrench produced by the contributions of all contact forces is larger. However, it does not take into account the shape of the ellipsoid, that is the relative contribution of each singular value. One of them could be very small, which would mean a grasp very close to lose the ability to withstand wrench in the associated direction.

- The grasp isotropy index is defined as [88], with σ_i the singular values of \mathbf{G} numbered in descending order:

$$Q_{GII} = \frac{\sigma_6}{\sigma_1} \quad (2.85)$$

This metric is an evaluation of the shape of the admissible object wrench ellipsoid: if the ellipsoid is a sphere, the index is maximal, and equal to 1. However, it does not give any information on the scale of the ellipsoid: an uniformly weak grasp, that is with all singular values close to each other, but small, will have a high Q_{GII} value.

Using Geometric Relations Between Contact Positions The main idea is to compute geometric relations between the contact point positions on the boundaries of the object [86], these geometric relations being correlated with desirable grasp properties. The main metrics in this category, described further in the following, are: shape of the grasp polygon [88], area of the grasp polygon [89, 90] and distance between the centroid of the contact polygon and the object center of mass [91–93].

- To optimize a planar grasp, a pertinent criterion is to tend toward a uniform distribution of the contact point around the object boundaries [88]. This is the shape of the grasp polygon metric. It reaches its optimal value when the polygon formed by the contact points is regular. This metric has a simple interpretation, and is easy to compute. However, it is designed for planar grasp only, and scaling it to 3D grasp may lead to misleading results: for example, in the case of an elongated object like a pen, the optimal grasp according to this metric is a polygon around the object smaller dimension.

- The robustness of a three-finger grasp is linked to the area of the triangle formed by the three contact points: the larger the area is, the better the grasp is, both for 2D and 3D grasps [89, 90]. This metric is known as the area of the grasp polygon. The extension of this metric to grasp involving more than 3 contacts is not straightforward, and requires to project additional contact points on a contact plane formed by three chosen contacts in order to produce a meaningful metric.
- Minimizing the distance between the centroid of the contact polygon and the object center of mass allows to reduce the influence of inertia and gravity, thus increasing the stability of the grasp [91–93]. This metric has a simple physical meaning, and is also simple to compute, provided that the center of mass is known, which is not always the case. A drawback of this metric is its independence to the number of contacts.

Determining the frictional form-closure property Two examples of such metrics are margin of uncertainty in finger positions [94, 95], and independent contact regions [96, 97].

- Taking into account the uncertainties in finger positioning relies on the concept of contact space. The contact space is defined as the n dimensional space representing the possible contact positions of n contacts on the boundary of a 2D object. The subset of this space allowing to keep the internal object forces inside the friction cone is called the force-closure space. This space does not take into account the ability of the gripper to apply the required wrenches, thus it corresponds to the frictional form-closure given by Prattichizzo and Trinkle [37] and not to the force-closure property defined in subsection 2.2.1. To minimize the influence of an uncertainty on contact positions, a good metric is to maximize the distance between the contact points and the boundaries of the force closure space [94, 95]. This metric can be computationally costly due to the friction cone computation, and subject to uncertainties regarding the friction coefficient. Moreover, it is cumbersome to apply to non-polygonal 2D object, or 3D object, due to the high dimensionality and complexity of the associated contact space.
- The concept of independent contact region is built on the previous concept of force-closure space. The goal is to find a set of object surface regions such that having each contact belonging to each independent region enforces the frictional form-closure condition, independently of the exact contact position in it [96]. Maximizing the size of the independent contact regions produces a larger set of possible force-closure grasps. A possible metric is for example to maximize the size of the smallest independent contact region of the set [97]. However, this metric also requires computing the friction cone, and is not intended for 3D grasps.

Taking into Account Limits on Contact Wrench Magnitude The metrics presented above do not take into account an eventual limit on the contact wrench magnitudes

(due to a limit on the joint torques). Taking into account this limit leads to more complex metrics, often with a higher computational cost, but makes the metric more realistic. An example of such metrics is the largest-minimum resisted wrench [98]. This metric relies on the computation of the grasp wrench space, the set of all admissible object wrenches given a contact locations and constraints on the contact wrench magnitudes and directions (due to friction constraints). An other metric conceptually close to the latter is the reachable wrench space under uncertainties, which takes into account uncertainties on contact position and orientation when computing the grasp wrench space [99].

There are also metrics having a tasks oriented criterion: if the task is well known, the most demanding directions in term of disruptive wrenches are also known. This concept is known as the task wrench space [87]. The metric can then promote a grasp that resists greater wrenches in these specific directions.

Metrics Depending on gripper Configuration

This type of quality metrics considers information about the gripper configuration, that is its geometry and joint positions, to assess the quality of a grasp.

A first category of metrics uses algebraic properties of the gripper-object Jacobian $\mathbf{H} = (\mathbf{G}^\top)^+ \mathbf{J}_g \in \mathbb{R}^{6 \times n_{gg}}$ (with $(\mathbf{G}^\top)^+$ the pseudoinverse of \mathbf{G}^\top). The main principle is similar to the metrics associated with algebraic properties of the grasp map \mathbf{G} . Here, the goal is to assess a necessary condition for the manipulability of the grasp, that is the ability for the gripper to exert any motion on the object. The metrics applied on \mathbf{G} can be extended to \mathbf{H} , and keep similar interpretation:

- The distance to singular configuration Q_{DSC} (analogue of Q_{MSV} for \mathbf{G}). This metric tends to keep the gripper away from its singular configurations [100]. Maximizing this metric minimizes the maximum transmission ratio between \mathbf{w}_o and \mathbf{t}_h .
- The volume of the manipulability ellipsoid Q_{VME} [101] (equivalent to Q_{VEW} for \mathbf{G}). A larger metric value means that for the same joint velocities, a larger object twist can be obtained.
- The uniformity of transformation Q_{UOT} corresponds to the condition number of \mathbf{H} [102] (similarly to Q_{GI} for \mathbf{G}). The metric value is maximum when the joint velocity contributions to the object twist are the same. This indicates that the gripper has a good manipulation ability in this configuration, as it is able to move the object in any directions without requiring disproportionate displacement of any joints.

An other pertinent metric regarding gripper configuration is related to the position of the finger joints [86]. With this criterion, a good configuration is obtained when joints are as far as possible from their mechanical limits, that is as close as possible to the center of their ranges. The main advantage of this metric is that it has an easy interpretation and

metric types		examples	include necessary condition for force- closure	assess friction constraints	low computational cost	include necessary condition for manipulability
contact position	Grasp Map	minimum singular value [87], volume of the ellipsoid in the wrench space [87], grasp isotropy index [88]	yes	no	yes	no
	geometric relations	shape [88] and area [89] of the grasp polygon, distance with the center of mass [92]	no	no	yes	no
	limitations on contact forces	largest-minimum resisted wrench [98], reachable wrench space under uncertainties [99]	yes	yes	no	no
	frictional form-closure tests	margin of uncertainty in finger positions [95], independent contact regions [97]	yes	yes	no	no
gripper configuration	gripper- object Jacobian	distance to singular configuration [100], volume of the manipulability ellipsoid [101], uniformity of transformation [102]	yes	no	yes	yes
	hand kinematic constraints	position of the finger joints [86]	no	no	yes	no

Table 2.5 – Summary of the main existing categories of grasp quality metrics.

computation. However, even if the optimal configuration with regard to this metric has a good range of possible motion on every joint, it does not mean that it can effectively transmit these motions to the contact point and to the object.

Finally, in a similar way to metrics depending on contact point positions, metrics evaluating the task compatibility of the gripper configuration can be assessed through the gripper-object Jacobian matrix \mathbf{H} . Knowing the velocity and force requirements of the task, these metrics promote gripper configurations able to ensure the maximum wrench or twist response along these directions [86, 103].

2.2.3 Chosen Grasp Quality Metric

The diversity of existing grasp quality metrics having been presented, the rationale behind the metric choice for this work can be explained.

First, this work focus on grasping tasks with underactuated multifingered grippers. It is chosen to make this study object-centric and not make the assumption of a specific target task. Therefore, task oriented metrics are not selected. Then, the ability to perform fine in-hand object manipulation will not be considered. Metrics based on gripper configuration are not pertinent in this case. In particular, metrics related to the gripper-object Jacobian are not best suited for underactuated grippers. Indeed, these metrics produce non-zero values only if $\dim(\text{Im}(\mathbf{H})) = 6$, with $\mathbf{H} = (\mathbf{G}^\top)^+ \mathbf{J}_{ag}$ in the case of underactuated grippers. This is possible if $\dim(\text{Im}(\mathbf{J}_{ag})) \geq 6$, that is if the gripper has six actuated joints at least, and if the defective space is not too large. However, the goal of using underactuated mechanisms is to reduce the number of actuators: it is very common for underactuated grippers to have less than six actuators. Moreover, the defective space associated with the underactuated joints can prevent the grasp to be manipulable. Applied on such grippers, all metrics based on \mathbf{H} would give null values for every possible grasps, which is not informative.

Thus, the choice is restrained to metrics depending on the contact point positions on the object surface. More specifically, as force-closure is an important property to have for a grasping task, metrics that include an assessment of one of its necessary conditions are preferred. Among these, the ones which are suitable only for 2D objects, although often simpler to compute, are not ideal, as the approximation of a 2D grasp greatly restricts the grasping ability of the gripper. An other aspect that needs to be taken into account is the computational cost of the metric. It needs to handle in a reasonable time the high number of contact points generated by multifingered grippers, especially when performing power grasps. With regard to this, metrics that require the evaluation of the friction cone constraints have a higher computational cost than metrics based on the Grasp Map only. This evaluation can be even more complex in the underactuated case, as shown previously in the example. Moreover, the computation of the friction cone constraints assumes that the friction coefficient between the gripper and the object is known. In practice, it is generally unknown, and tedious to measure experimentally. With a wrong value, the risk is to compute a grasp quality that do not matches the behavior obtained in the real experiments, thus creating simulation-to-real issues.

Considering these elements, the choice is now only between metrics based on the

Grasp Map \mathbf{G} . They provide a necessary condition for force-closure by assessing if $\dim(\text{Im}(\mathbf{G})) = 6$. According to Mnyussiwalla et al. [104], the values of the three metrics are highly correlated for grasps with three contacts (in particular, Q_{MSV} and Q_{GII} even have a correlation coefficient of 1). Thus, even if they have slightly different interpretations, they in fact measure the same physical property. Among them, the minimal singular value of \mathbf{G} , Q_{MSV} , has been chosen. Indeed, it ranks grasps according to how far they are from losing the ability to resist wrench in the most unfavourable direction (worst case scenario). It provides a simple and robust information, while guaranteeing that force-closure may be achievable in this configuration.

2.3 Conclusion

This chapter has presented the formalisms regarding grippers and grasps modeling that are required to better understand this work. More specifically, the influence of the underactuation on this modeling has been explained.

Moreover, some important properties of a grasp have been highlighted. These properties determine to what extent a given grasp configuration allows a gripper to enforce a given twist or wrench to the object, or allows it to maintain friction constraints. These properties have been derived in the specific case of underactuation, to show the effect of this mechanism on the grasping ability of a gripper. Various quality metrics, derived from these properties, have also been presented.

Finally, the metric used to rank grasps in this work, the minimal singular value of \mathbf{G} , Q_{MSV} , has been introduced, and the various factors justifying this choice have been presented.

Chapter 3

Human Initiated Grasp Space Exploration Approach

The grasp space exploration phase is one of the most critical component in the versatile grasp planner framework. The efficiency and quality of the grasps found in this phase will determine the ones of the grasps generated in the planning phase. The specificity of underactuated grippers, introduced in chapter 1 and further detailed in chapter 2, makes this phase more challenging than for fully actuated grippers or simpler grippers. In this chapter, a method to efficiently explore the grasp space of an underactuated gripper is proposed. First, its main hypotheses are explained. Then, this method is presented. It relies on a generative model to learn a compact representation of the grasp space.

Contents

3.1	Problem Statement	90
3.1.1	Considered Hypotheses	90
3.1.2	Input & Output Data	92
3.2	Techniques for Data Generation & Dimensionality Reduction	95
3.2.1	Dimensionality Reduction Techniques	95
3.2.2	Generative Models	104
3.2.3	Chosen Approach	107
3.3	Variational Auto-Encoder: Principle and Techniques	107
3.3.1	Mathematical Principles	107
3.3.2	Training and Generating Samples	110
3.3.3	VAE Variants	112
3.4	Variational Auto-Encoders for Grasp Space Exploration	114
3.4.1	Primitives Dataset	116
3.4.2	Human-initiated Grasp Generator (HGG)	117
3.4.3	Dataset Extension & Grasp Quality Computation	117
3.4.4	Quality-oriented Grasp Generator (QGG)	118
3.5	Conclusion	120

3.1 Problem Statement

An essential step of any grasp planning algorithm is the grasp space exploration phase. This phase, as well as the grasp planning as a whole, depends highly on the mechanical architecture and kinematics of the considered gripper.

3.1.1 Considered Hypotheses

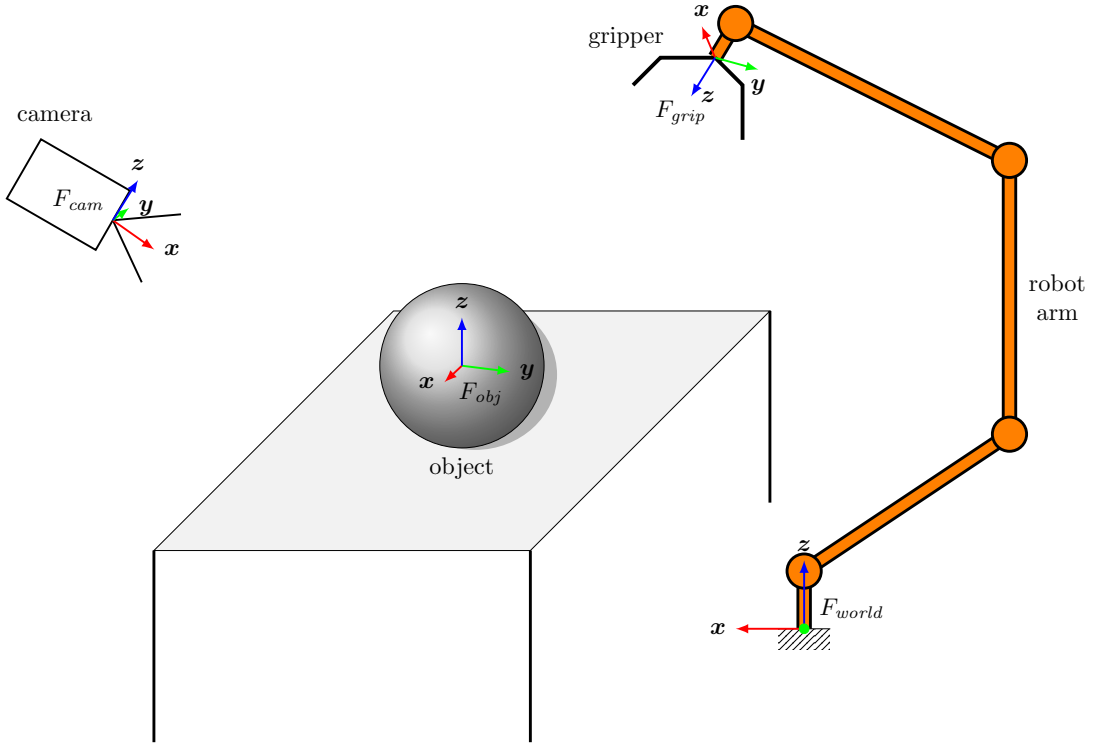


Figure 3.1 – Scheme of the considered setup and its associated frames.

As stated in subsection 1.4.2 and further highlighted in chapter 2, underactuated grippers have complex behavior that can make difficult or impossible to control the contact point locations on the object surface. Indeed, the uncontrollable torques produced by the passive springs of the underactuation system makes the contact point positions dependent on the force equilibrium between the gripper and the object. In this conditions, a grasp space exploration approach based on contact point positions is not suitable. Hence, in this work, a gripper configuration approach is chosen, where the grasp space is explored by testing different gripper configurations. As stated in subsection 1.4.1, the gripper configuration space (of dimension d_{conf}) is constituted of the spatial configuration space (of dimension d_{space}) plus the gripper internal configuration space (of dimension d_{int}). The gripper spatial configuration space is chosen equal to the Euclidean space $SE(3)$. It allows to fully leverage the grasping ability and kinematic potential of the gripper, as

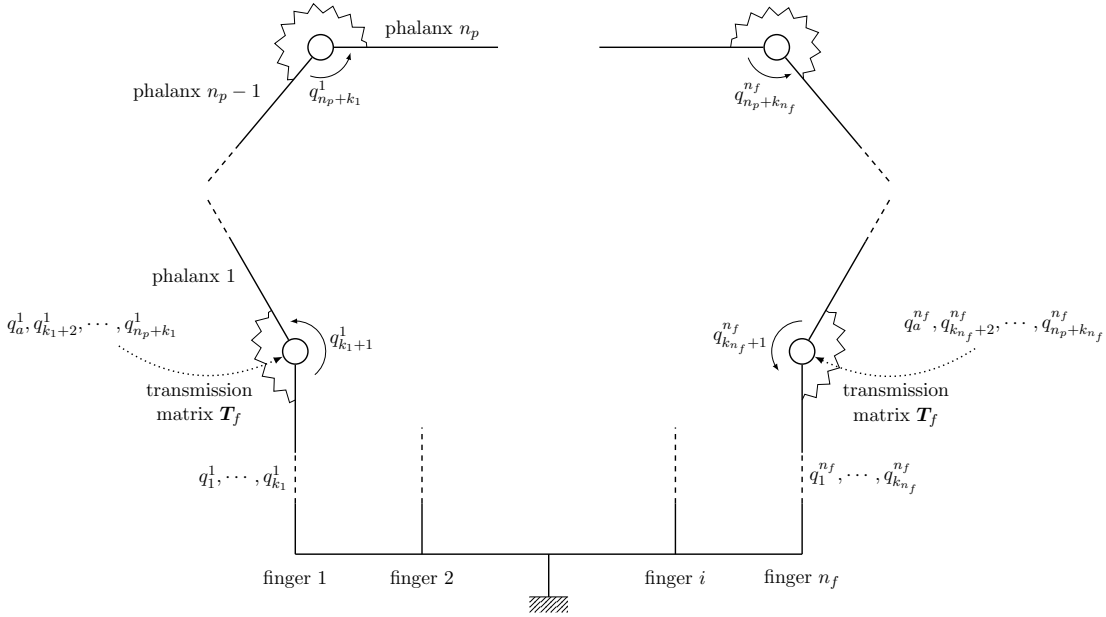


Figure 3.2 – Scheme of the considered underactuated gripper. This gripper has n_f underactuated fingers. Each finger has n_p underactuated phalanges, and k_i fully-actuated degrees of freedom allowing finger repositioning relative to the palm (which can include abduction-adduction motion or any other degrees of freedom).

opposed for example to a planar spatial configuration space. The choice of parameters for the gripper internal configuration space is further explained in the following.

The considered robotic setup, displayed in Figure 3.1, is constituted of an underactuated gripper mounted on the end-effector of a robotic arm manipulator. To achieve an arbitrary gripper pose in the Euclidean space $SE(3)$, the robotic manipulator needs to have at least six degrees of freedom. The position of the gripper is parameterized through a frame F_{grip} attached to its palm.

For its part, the gripper has $n_f > 2$ fingers attached to a palm. It is assumed that each finger has the same phalanx architecture and the same number n_p of phalanges and underactuated joints. A scheme of the considered gripper architecture is displayed in Figure 3.2. Each finger has k_i actuated joints at its base (with i the index of the finger), able to reorient it relative to the palm and to the other fingers. These repositioning degrees of freedom ($q_1^i, \dots, q_{k_i}^i$) can be translational or rotational, and can include abduction-adduction joints. They constitute the gripper internal configuration space. The individual positions of the finger underactuated joints are not considered in the internal configuration space: the fingers start fully opened, and they are closed simultaneously to grasp the object, with the same predetermined target actuator position. Indeed, the positions of the finger underactuated joints are not individually controllable: they are only related to the actuator position through the transmission matrix T_f , as shown in chapter 2. Thus, they depend on the final force equilibrium between the gripper

and the object, as the contact point positions: each underactuated joint reaches a position where its passive spring produces the required torque for static equilibrium. As a given actuator position can produce an infinity of finger configurations (depending on the object shape and position relative to the gripper), it would not be pertinent to include the finger actuator positions in the gripper internal configuration. Thus, the considered gripper configuration space dimension is:

$$d_{conf} = d_{space} + d_{int} = 6 + \sum_{i=1}^{n_f} k_i \quad (3.1)$$

An object is supposed to be placed on a plan in the workspace of the considered robotic setup, as shown in Figure 3.1. It is assumed that the object geometry is known a priori, as well as the pose in the scene of its associated frame F_{obj} . The object pose can be known thanks to exteroceptive sensors and object pose estimation algorithms [31, 58–60]: the pose of the object frame F_{obj} is retrieved and expressed in the camera frame F_{cam} . Then, knowing the pose of the camera frame in the world frame F_{world} , one can express the object frame in it.

Considering the surface holding the object allows to take into account the geometric and kinematic constraints that it generates during the grasp space exploration procedure. Indeed, this surface can prevent many gripper configurations due to the collision of the gripper with it. Methods considering a free floating object during the grasp space exploration exist, but as a result they can generate a significant number of gripper configurations that need to be eliminated afterwards.

3.1.2 Input & Output Data

The goal of the grasp space exploration is to build a collection of diversified grasps, so that it is possible to propose an appropriate grasp to the trajectory planner and controller, that is with the best quality possible, for as many possible tasks as possible. The goal of the presented method is to generate gripper configurations with a prediction of their grasp quality so that such collection can be built and used efficiently.

Outputs

Our method has two outputs:

- The gripper configuration \mathbf{g} , composed of two parts:
 - The gripper spatial configuration, expressed as the pose of the gripper frame F_{grip} in the object frame F_{obj} in order to be invariant to object pose:

$$(x, y, z, q_x, q_y, q_z, q_w) \in \mathbb{R}^3 \times \text{Sp}(1) = \text{SE}(3) \quad (3.2)$$

with x, y, z the Cartesian position of the frame, and q_x, q_y, q_z, q_w its orientation expressed in quaternion convention. The quaternion representation is chosen for its compactness and easier interpolation compared to matrix representation, and to avoid singularities that may arise when using Euler convention.

- The gripper internal configuration space,

$$\left(q_1^1, \dots, q_{k_1}^1, \dots, q_1^{n_f}, \dots, q_{k_{n_f}}^{n_f} \right) \in \prod_{i=1}^{n_f} \left(\prod_{j=1}^{k_i} [q_{j,\min}^i, q_{j,\max}^i] \right) \quad (3.3)$$

with $q_{j,\min}^i$ and $q_{j,\max}^i$ respectively the minimum and maximum joint limit of the j^{th} repositioning joint of the i^{th} finger, and \times the Cartesian product.

Thus, the gripper configuration space has d_{conf} dimensions, but a gripper configuration is described by $d_{conf} + 1$ non-independent variables due to the choice of a quaternion representation for the rotation.

- a prediction \widehat{Q}_{MSV} of the quality value Q_{MSV} (as described in Equation 2.83) of the grasp produced by this configuration, if any.

Inputs

As explained in the previous subsection, this work takes into account the surface on which the object is lying. The pose of the object relative to this surface determines partially the area of the grasp space that are accessible due to geometric and kinematic constraints. Thus, an input of the grasp exploration should allow to identify in which stable pose the object is resting, in order to explore only gripper configurations suitable for it.



Definition 7 (Stable pose). A stable pose of an object is one of the geometrically distinct pose on which it can rest at static equilibrium when lying on an horizontal surface. In this work, the stable poses are identified thanks to the vector $e = (a, b, c, d)$ of the Cartesian equation of the tabletop plane on which the object is lying, expressed in the object frame F_{obj} . The vector (a, b, c) is the tabletop plane normal vector, and d is the distance between the frame F_{obj} origin and its projection on the tabletop plane. In case several representations exist for the same stable pose, one of them is chosen arbitrarily. It can happen when the object has symmetries that create several geometrically equivalent poses corresponding to the same stable pose. An example is shown for a right cuboid in Figure 3.3.

Grasp Generation Issue

The chosen gripper configuration space is of high dimensions ($d_{conf} = 6 + \sum_i^{n_f} k_i$). It is unlikely that configurations simply generated by sampling values in this huge space belong to the grasp space, because of the numerous constraints the grasp space is subject to (namely geometric, static and kinematic constraints), as explained in section 1.4. However, as mentioned in the same section, and illustrated in Figure 1.15, the grasp

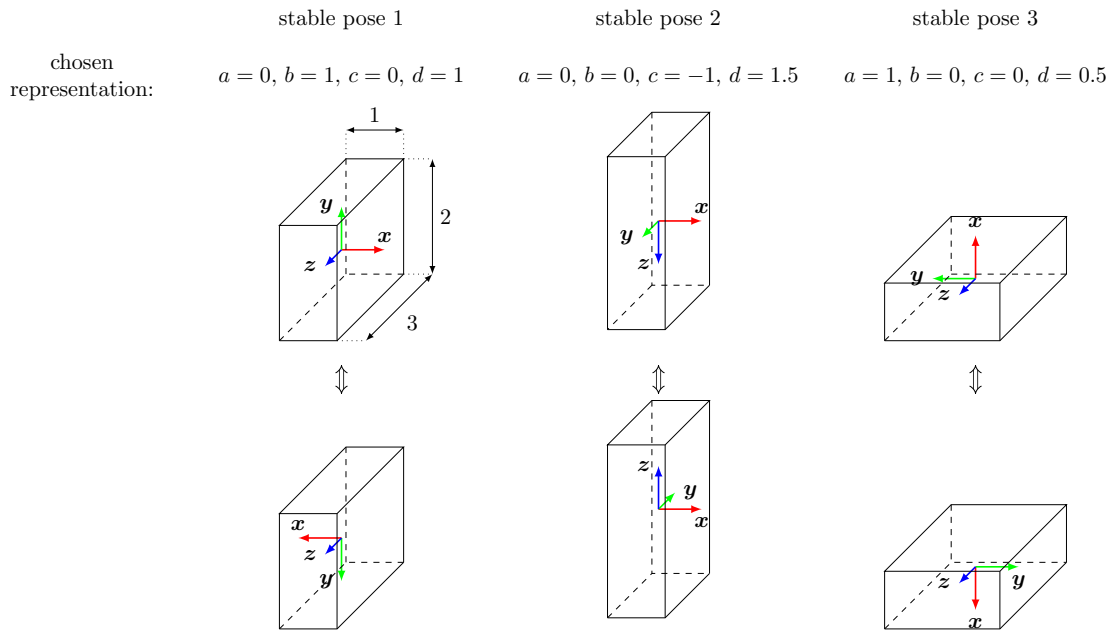


Figure 3.3 – The three stable poses of a right cuboid. For each stable pose, the two rows are two geometrically equivalent poses, that are due to symmetries in the object geometry. The orientation of the top row is chosen to express the Cartesian equation representing each stable pose.

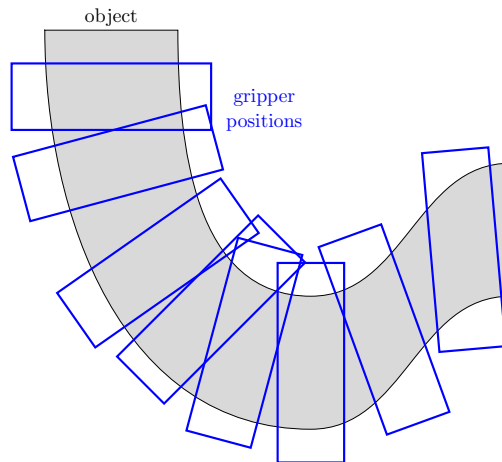


Figure 3.4 – Examples of successful grasps on an arbitrary 2D object, in the simple case of a bi-digital gripper with parallel jaws. Here, $d_{conf} = 3$, and the gripper configuration space is parameterized by three variables: (x, y, θ) .

space can probably be thought of as a set of submanifolds of the gripper configuration space. To further illustrate this idea, some examples of successful grasps for a bi-digital

parallel gripper on a 2D object is displayed in Figure 3.4. In this example, it is clear that the x , y and θ values of successful grasps are highly correlated, and that they only cover a very small part of the gripper configuration space. In this example, it is very likely that the grasp space is in a one or two dimensional submanifold of the gripper configuration space.

Thus, it is possible to assume that the grasp space can be parameterized by a set of unobserved latent variables, which is smaller than the set of parameters describing the gripper configuration space. Discovering these latent variables should allow to have an insight on the structure of the grasp space, and ultimately to generate gripper configurations belonging to it much more reliably. This is the approach chosen in this work, and it requires two main abilities:

- being able to reduce the dimension of the gripper configuration space to find the latent variables at the origin of the grasp space,
- being able to reliably generate new grasps from this compressed representation.

3.2 Techniques for Data Generation & Dimensionality Reduction

Several techniques have been developed to reduce the dimensionality of the feature space of a given dataset, or to generate new data resembling it. An overview of some of the main existing techniques is given in the following.



Definition 8 (Dimensionality reduction). It is the transformation of data from a high-dimensional space into a low-dimensional space so that the low-dimensional representation retains some meaningful properties of the original data, ideally close to its intrinsic dimension.

Definition 9 (Generative model). A generative model makes the assumption that the observed data are some samples of a vector of random variables \mathbf{x} following an unknown distribution $P(\mathbf{x})$. It is assumed that there is also a vector of unobserved (latent) random variables \mathbf{z} following a distribution $P(\mathbf{z})$. The goal is to build a model of the conditional probability distribution of \mathbf{x} given a sample of \mathbf{z} , $P(\mathbf{x}|\mathbf{z})$. Sampling from this distribution allows to generate new samples of \mathbf{x} .

3.2.1 Dimensionality Reduction Techniques

Dimensionality reduction techniques aim at projecting a high dimensional input space into a reduced set of features. They are able to create new compressed features from the

initial set of features. These new features are more representative of the data variability than the initial ones and encapsulate them. This transformation is done thanks to algebraic transformation of the input data, and following a given optimization criterion. Dimensionality reduction techniques can be mainly divided in two categories, linear and non-linear, based on the type of transformation they use.

Linear Methods

Principal Component Analysis One of the most commonly used linear dimensionality reduction technique is the Principal Component Analysis (PCA) [105]. The goal is to find orthogonal directions in the initial space of m features that explain as much data variance as possible, these directions being called principal components. This method is based on algebraic properties of $\mathbf{X} \in \mathbb{R}^{n \times m}$ the centered matrix representing the dataset of n samples in the feature space of m dimensions. The transformation applied by the PCA corresponds to the singular value decomposition of \mathbf{X} , or equivalently to the diagonalization of the empirical sample covariance matrix $\mathbf{Q} \in \mathbb{R}^{n \times n}$. The empirical sample covariance matrix of the data matrix \mathbf{X}^\top is proportional to $\mathbf{X}^\top \mathbf{X}$, as $\mathbf{Q} = \frac{1}{n-1} \mathbf{X}^\top \mathbf{X}$. Here, the principle is described with the diagonalization of \mathbf{Q} , but most PCA implementation use the singular value decomposition for efficiency. Diagonalizing \mathbf{Q} gives:

$$\mathbf{Q} \propto \mathbf{X}^\top \mathbf{X} = \mathbf{W} \mathbf{\Lambda} \mathbf{W}^\top \quad (3.4)$$

with $\mathbf{W} \in \mathbb{R}^{m \times m}$ a matrix whose columns are the eigenvectors of $\mathbf{X}^\top \mathbf{X}$ (or equivalently the right singular vectors of \mathbf{X}), and $\mathbf{\Lambda}$ the diagonal matrix of eigenvalues of $\mathbf{X}^\top \mathbf{X}$ (or equivalently of squared singular values of \mathbf{X}). The value of each eigenvalue correspond to the data variance explained by the corresponding eigenvector, ranked in decreasing order.

Thus, the matrix \mathbf{W} is a transformation matrix that allows to express \mathbf{X} in an orthogonal base, or equivalently an uncorrelated base, formed by the principal components, with the explained variance maximized on each axis:

$$\mathbf{T} = \mathbf{X} \mathbf{W} \quad (3.5)$$

with $\mathbf{T} \in \mathbb{R}^{n \times m}$ the transformed data matrix.

However, as the eigenvectors corresponding to the higher eigenvalues explain the most data variance, one does not necessarily need to keep every principal component of \mathbf{W} . A dimensionality reduction can be obtained by choosing a matrix $\mathbf{W}_f \in \mathbb{R}^{m \times f}$ with $f < m$, selecting only the eigenvectors corresponding to the f higher eigenvalues. The previous equation becomes:

$$\mathbf{T}_f = \mathbf{X} \mathbf{W}_f \quad (3.6)$$

Here, the dimension of \mathbf{T}_f is $\mathbb{R}^{n \times f}$: the feature dimension has been reduced after the transformation. As the preserved dimensions maximize the explained variance, \mathbf{T}_f minimizes the total squared reconstruction error ε for any choice of f :

$$\varepsilon = \|\mathbf{X} - \mathbf{T}_f \mathbf{W}_f^\top\|_2^2 = \|\mathbf{X} - \mathbf{X}_f\|_2^2 \quad (3.7)$$

Metric Multidimensional Scaling (MDS) It aims at finding a low dimensional embedding that best preserves the pairwise distance between samples of the dataset [106]. It is based on algebraic properties of the matrix of similarity between samples, $\mathbf{B} \in \mathbb{R}^{n \times n}$. Thus, knowing explicitly the sample matrix \mathbf{X} is not required: only a distance between samples is needed. If the metric used in the similarity matrix \mathbf{B} is the Euclidean distance between samples, MDS applied on \mathbf{B} gives the same transformation as a PCA applied on \mathbf{X} . In the general case, any distance can be used and not only the Euclidean distance. MDS is mostly used to project in a two or three dimensional space for data visualisation.

These dimensionality reduction techniques can have multiple applications. First, reducing a high dimensional dataset on two dimensions allows to display it in a meaningful way, in order to visualize if clusters exist in the dataset. More generally, inspecting the explained variances of the principal components in a scree plot allows to determine the number of pertinent dimensions of a given dataset. It can also be useful for denoising purpose, the last eigenvectors being dominated by the dataset noise. Finally, multiple machine learning algorithms can benefit from a reduced input dimensionality.

The PCA or MDS are not generative, that is, it is not possible to generate new data reliably. First, MDS uses the similarity matrix \mathbf{B} as input, and not the data matrix \mathbf{X} itself. As it works only with the relative positions between data points, it is impossible to produce an inverse transformation that outputs a data point. Regarding PCA, an inverse transformation exists, but it can only transform back from the transformed space points that are very close to dataset points, and only if not too much compression occurred. Indeed, an arbitrary vector $\mathbf{t} \in \mathbb{R}^m$ lies most likely far from any transformed dataset points. Thus, applying the inverse transformation $\mathbf{t}\mathbf{W}^\top = \mathbf{x}_{gen}$ will produce an arbitrary vector $\mathbf{x}_{gen} \in \mathbb{R}^m$, which will most likely not be consistent with the original data contained in \mathbf{X} .

The main limitation of these methods is that they can only extract linear relationships between the features of the dataset. For example, if the dataset lies in a manifold embedded in the initial space, they will fail to retrieve this information. It will simply construct the best possible linear subspace maximizing the explained variance for PCA, or preserving the distance for MDS. For example, if the data is following a one dimensional circular shape in the initial feature space, they will only be able to reduce the dimension to the two dimensional plane in which the data lies, and not to the one dimensional manifold.

Non-Linear Methods

To be able to take into account non linear relationships in the dimensionality reduction, non-linear methods have been developed.

Kernel-PCA A first example is an extension of PCA to the non-linear case, the kernel PCA [107]. The main idea behind kernel PCA is to project the dataset \mathbf{X} into a very

high dimensional feature space thanks to an arbitrary function $\Phi : \mathbb{R}^m \rightarrow \mathbb{R}^M$, $m < M$. Applying the PCA in this feature space is equivalent to finding non-linear relationships in the initial data space. In the feature space, the covariance matrix becomes:

$$\mathbf{Q} \propto \Phi(\mathbf{X})^\top \Phi(\mathbf{X}) \quad (3.8)$$

Performing the PCA in the feature space requires to diagonalize this matrix. However, Φ is an arbitrary high dimensional function, and it can be intractable to compute $\Phi(\mathbf{X})$. To avoid this computation, the kernel trick can be used, as in Support Vector Machine [108, 109]. This trick relies on the fact that a kernel function \mathbf{K} can be chosen so that there exists a function Φ following the relation [110]:

$$\mathbf{K}(\mathbf{X}) = \Phi(\mathbf{X})^\top \Phi(\mathbf{X}) \quad (3.9)$$

With this trick, one does not need to explicitly compute Φ to perform the PCA. Only the diagonalization of \mathbf{K} is required. More in depth mathematical developments are given in Schölkopf et al. [107].

Compared to classic PCA, kernel PCA computes the projection of the data samples on the principal components (that is the eigenvectors of \mathbf{K}), and not the components themselves. An other difference is that the eigenvalues of \mathbf{K} are not related to the explained variance of the principal components, and cannot be used in the same way as classic PCA to select the number of dimensions to keep. Finally, due to the projection of the data samples in a high dimensional feature space through the function Φ , finding the inverse transformation is not straightforward: several techniques were developed to find the pre-image in the initial space of a point in the feature space, for example in Bakır et al. [111]. Even when the reconstruction of a feature space point in the initial space is possible, there is an issue similar to classic PCA when trying to generate new data: it is non-trivial to find a point in feature space that will produce a data point consistent with the original ones.

Locally Linear Embedding Another non-linear dimensionality reduction method is the Locally Linear Embedding (LLE) [112]. Let $\mathbf{X} \in \mathbb{R}^{n \times m}$ a data matrix of n samples, each of dimensionality m , sampled from a manifold. The manifold structure allows to assume that each data sample and its neighborhood lies in a locally linear area, provided that the neighborhood size is sufficiently small. Thus, each data sample can be linearly expressed from its neighbors. The relation between each data sample and its neighbors can be stored in a matrix $\mathbf{W} \in \mathbb{R}^{n \times n}$. The matrix \mathbf{W} characterize intrinsic geometric properties regarding the neighborhood of each data samples. If the data are sampled from a manifold of dimensionality $f < m$, there exists a linear transformation that allows to map the m coordinates of each data sample in the initial space to a new f dimensional coordinates system on the manifold. The data matrix \mathbf{X} is mapped to a transformed data matrix $\mathbf{T} \in \mathbb{R}^{n \times f}$, using the knowledge from the matrix \mathbf{W} . One of the interesting feature of LLE is that by computing both local (through the \mathbf{W} matrix) and global (through the \mathbf{T} matrix) relations, it is able to capture non-linear relationships between the data samples, while performing only linear computations.

Isometric Feature Mapping Another example of non-linear dimensionality reduction method is isometric feature mapping, or isomap [113]. This technique can be viewed as an extension of metric MDS using geodesic distances, which enables it to capture the non-linear structure of a dataset, contrary to the classic Euclidean distance for example. In a similar way to LLE, isomap merges local information (used to compute geodesic distances) and global information (the geodesic distances themselves) to represent non-linear relationships in the dataset.

These algorithms are very effective to create human readable representation in low dimensions of a dataset having non-linearity, or to preprocess such dataset for a downstream machine learning algorithm, for classification or clustering for example. However, as for kernel PCA, it is not straightforward to select the number of dimensions on which the projection is performed. Moreover, such methods are unable to reconstruct a point in the initial space from a point in the reduced space, and thus they cannot generate new data.

Autoencoders One of the most flexible and powerful techniques for non linear dimensionality reduction is autoencoders. The first occurrences and descriptions of this concept can be found in Ballard [114], Lecun [115], Bourlard and Kamp [116], and Hinton and Zemel [117]. The power and flexibility of this technique comes from the fact that it relies on neural networks to learn a compact representation of an input dataset. It is composed of two main parts: an encoder, and a decoder. A scheme of the general architecture of this method is displayed In Figure 3.5.

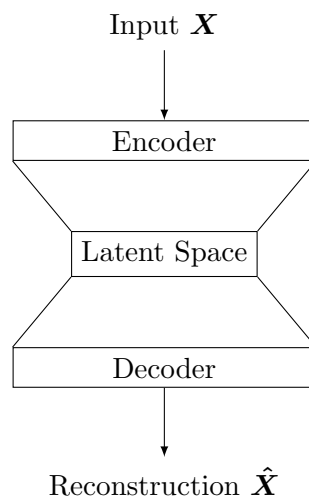


Figure 3.5 – Structure of an autoencoder.

The goal of an autoencoder is to reconstruct its input \mathbf{X} at its output, using a compressed representation of the data in the latent space [118]. The reconstruction is denoted $\hat{\mathbf{X}}$. The “hat” notation is to emphasize that it is an estimation made by the

autoencoder, the goal being to have $\mathbf{X} = \hat{\mathbf{X}}$. The encoder and decoder are usually implemented through feedforward neural networks. The encoder inputs are the rows of the matrix \mathbf{X} , and its outputs are the corresponding values of the latent variables in the latent space. A data compression occurs if the latent space dimension is smaller than the number of features of \mathbf{X} . The decoder inputs are the latent variable values, and its outputs are the rows of the matrix $\hat{\mathbf{X}}$, the reconstruction of the encoder inputs.



Definition 10 (Artificial neuron). A neuron is an elementary component of a neural network. It computes its output value y , also called *activation*, from its input vector $\mathbf{x} \in \mathbb{R}^p$ as follows:

$$y(\mathbf{x}) = a \left(\sum_{i=1}^p w_i x_i + b \right) \quad (3.10)$$

with w_i the neuron *weight* corresponding to the i^{th} component of its input, b the neuron *bias*, and a a function called the *activation function* of the neuron, which is often a non linear function.

Feedforward neural networks are constituted of several layers of artificial neurons stacked on each other, the j^{th} layer having l_j neurons. The output $\mathbf{y}_j \in \mathbb{R}^{l_j}$ of layer j becomes the input \mathbf{x}_{j+1} of layer $j + 1$. For the first layer, $\mathbf{x}_1 \in \mathbb{R}^m$ corresponds to a row \mathbf{X}_k of the data matrix \mathbf{X} . For the j^{th} layer of l_j neurons, the output \mathbf{y}_j can be computed as follows:

$$\mathbf{y}_j(\mathbf{x}_j) = a(\mathbf{W}_j \mathbf{x}_j + \mathbf{b}_j) \quad (3.11)$$

with $\mathbf{W}_j \in \mathbb{R}^{l_j \times l_{j-1}}$ the layer weight matrix, $\mathbf{b}_j \in \mathbb{R}^{l_j}$ the layer bias vector, and a being applied componentwise. This matrix representation is very useful for the implementation of neural networks. A scheme of a simple network is shown in Figure 3.6. In an autoencoder, the latent space is in practice the output of the last layer of the encoder. If this layer has less neurons than the number of features of the dataset, m , the information is compressed.

The activation function allows to introduce non linearity in the neural network. Without it, a neural network outputs only a linear combination of its inputs. Several activation functions have been described in the literature, the three mainly used being the sigmoid function a_{sigm} , the hyperbolic tangent function a_{tanh} , and the rectified linear unit function a_{relu} . To be used as activation function, a function must necessarily be differentiable, or at least admit a left and right derivative at any point. In general, hyperbolic tangent is preferred over sigmoid function because it allows a faster convergence [119]. It has been demonstrated experimentally that for deep networks, the rectified linear unit perform better than hyperbolic tangent [120]. Their graphs are in Figure 3.7 and their

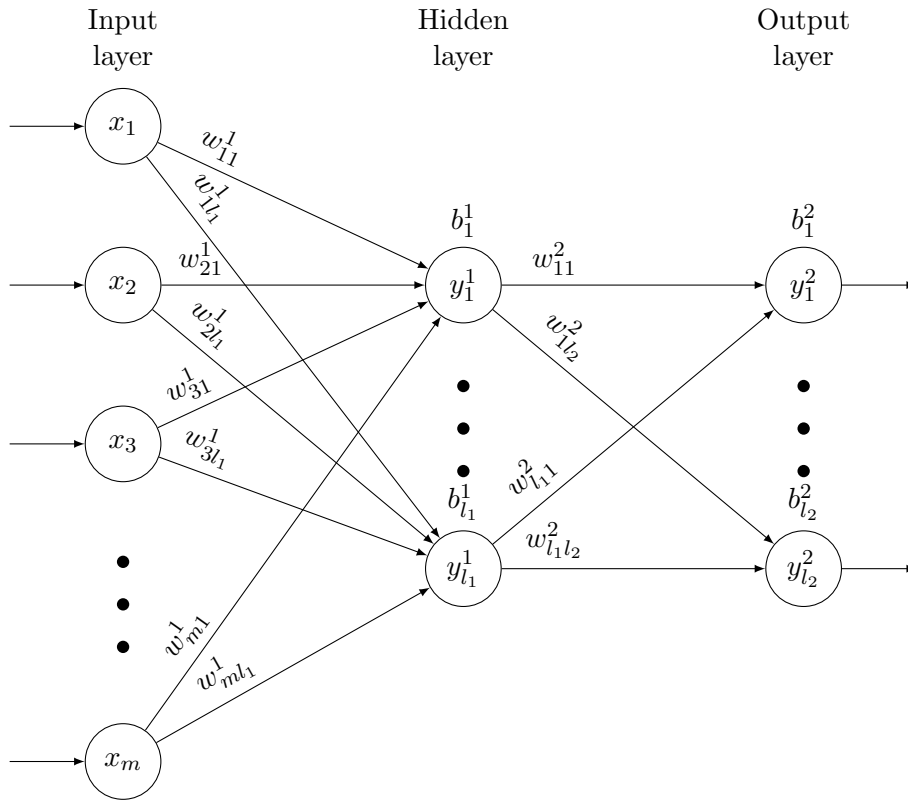


Figure 3.6 – Scheme of a feedforward neural network with one hidden layer.

expressions are as follows:

$$a_{\text{sigm}}(z) = \frac{1}{1 + e^{-z}} \quad a_{\text{tanh}}(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}} \quad a_{\text{relu}}(z) = \max(0, z) \quad (3.12)$$

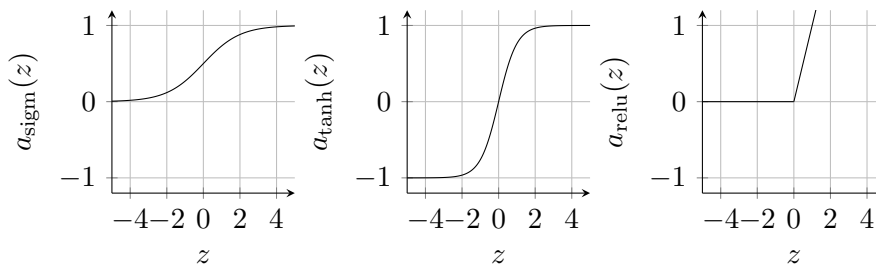


Figure 3.7 – Graphs of the three main activation functions.

The weights and biases of the neurons are adjustable parameters, that are optimized iteratively usually with stochastic gradient descent [121] during a learning phase, the gradient of a chosen cost function being computed thanks to the back-propagation algorithm [122]. The cost function can be for example the mean squared reconstruction

error:

$$\varepsilon = \frac{1}{n} \sum_{k=1}^n \|\mathbf{X}_k - \hat{\mathbf{X}}_k\|_2^2 \quad (3.13)$$

The back-propagation method relies on the computation of the partial derivatives of the cost function with respect to weights and biases. It aims to propagate backward the gradient descent to the weights and biases of each neuron of each layer, to update them before the next gradient descent iteration. The gradient descent step size is called the learning rate. More in depth mathematical development are available in Goodfellow et al. [118] and Nielsen [123]. Several implementation of gradient descent have been used to optimize neural networks [124]. They are mainly based on the Stochastic Gradient Descent method, which is an optimization technique not limited to the field of neural networks [125]. Variant of this algorithm have been proposed recently by the neural networks community to improve its performances, as for example adaptive learning rate methods like RMSProp [126] or Adam [127] algorithms. The main idea of these methods is to change the learning rate depending on previous values of the gradient.

Neural networks are a very powerful tool: by combining and chaining simple linear operations with easy to compute non-linear functions, the whole network can represent complex functions.



Theorem 1 (Universal approximation theorem). *A feedforward network with a linear output layer (with a linear activation function) and at least one non-linear hidden layer (with a non-linear activation function) can approximate any continuous function on a closed and bounded subset of a finite-dimensional space, with an arbitrarily small amount of error if the network has enough neurons in the hidden layers [118].*

For more precise and in depth mathematical formulation, see Hornik et al. [128] and Cybenko [129].

This theorem guarantees that a neural network representing a given function exists, but it gives no information on the ability of the gradient descent algorithm to find the correct weights and biases corresponding to it for a given network architecture. Indeed, the main drawback of the gradient descent is that there is no guarantee to find the global minimum: it can get stuck in a local minimum, which leads to sub-optimal performances.

During its training, a neural network can encounter two main setbacks: underfitting and overfitting.



Definition 11 (Underfitting). It occurs when the network does not reach sufficiently high performances on the training data (measured for example with the cost function value).

Definition 12 (Overfitting). It occurs when there is a too important performance gap between the performances obtained on training data and the ones obtained on unseen data. The performance of a network on unseen data is also called the generalization error.

One way to prevent underfitting is to increase the capacity of the network, that is its number of trainable parameters. To limit overfitting, several methods have been developed, called regularization [118]. These methods can for example set penalties on the trainable parameters norm (L1 or L2 regularizer, also known as weight decay), end the learning procedure when the performances stop improving during the training process (early stopping), or prevent the neurons to depend heavily on each other, by randomly removing some of them during the training (as the dropout method for example).

The autoencoder has several advantages over the other presented dimensionality reduction techniques which are mainly due to the neural network framework. Firstly, it allows to design specialized layers that can take into account a priori knowledge about the data, as for example Convolutional Neural Networks for image processing [130]. More generally, the modularity of the neural network framework allows to adapt its layers and their connectivity to the issue at hand. Moreover, the universal approximation theorem allows an autoencoder, at least theoretically, to both compress and reconstruct any type of input data, with any compression rate. Indeed, for a given set of data samples, there exist a line that pass exactly through all data points. This function can be represented with a neural network. However, it is worth noting that the produced network will reproduce exactly the training data, but will be unable to generalize to new data: it will be overfitting. To avoid this, regularization techniques can be used to force the network to extract only the most pertinent information from the data at the expense of the reconstruction error, or an appropriate compression rate can be chosen from a priori knowledge on the data.

The data mapping in the latent space of an autoencoder can have an arbitrary internal structure: the gradient descent will converge toward the most efficient way to reconstruct the input data, regardless where the data are mapped in latent space. Thus, areas of the latent space that correspond to consistent data can be very sparse, and areas that are far from any data samples are most likely meaningless. Hence, it is not possible to generate easily new data with an autoencoder.

3.2.2 Generative Models

The primary goal of generative models is to extract information from the dataset so that they can generate new samples that could have been in the original dataset. For that, they can infer the latent variables that describes the underlying structure of the data, but it is not an explicit goal, contrary to the dimensionality reduction techniques. They can rely on an explicit probabilistic modeling as stated in Definition 9, but it is not necessarily the case (for example in Generative Adversarial Network, a model described later, the probabilistic modeling is only implicit).

In recent years, generative models received a lot of interest, mainly for their potential use in image and natural language processing [131]. Most of recent works dealing with generative models are based on the use of various types of neural networks.

First attempts to create generative models involved stochastic neural networks, which are different from feedforward neural networks presented previously. Such techniques include for example Restricted Boltzmann Machine [132], Deep Boltzmann Machine [133], or Deep Belief Network [134]. However, these techniques result in blurry reconstruction of the target sample [131], thus they are not suitable when a sharp reconstruction is required, for example in image generation. Moreover, these models need to be designed carefully, as they require multiple properties to be ensured to maintain tractability [118].

Variational Auto-Encoder



Definition 13 (Variational Auto-Encoder (VAE)). It is a generative model (Definition 9) introduced by Kingma and Welling [135]. It is based on neural networks, and has an architecture close to classic autoencoders. It aims at maximizing the probability of generating samples corresponding to the ones in the dataset. For that, it approximates during its training the two unknown conditional distributions $P(\mathbf{z}|\mathbf{x})$ (the encoder), and $P(\mathbf{x}|\mathbf{z})$ (the decoder), with the assumption that $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. More in depth mathematical developments on this method are given in section 3.3.

Recently, a generative model based on the autoencoder framework has been developed: the Variational Auto-Encoder (VAE). To generate data from latent variables more consistently than with classic autoencoder, some assumptions are made on the structure of the encoder, latent space and decoder. The main advantage is that the standard normal assumption for the latent space ($\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$) forces it to be relatively smooth, which prevents overfitting to some extent. It also pushes the VAE to learn a dense and disentangled representation in its latent space, and encourages it to uses as few dimensions as possible. Moreover, learning to simultaneously encode and decode this representation forces the VAE to create a predictable and structured coordinate system in the latent space. This makes the VAE particularly effective to capture low-dimensional manifolds

embedded in the data [118].

Generative Adversarial Network

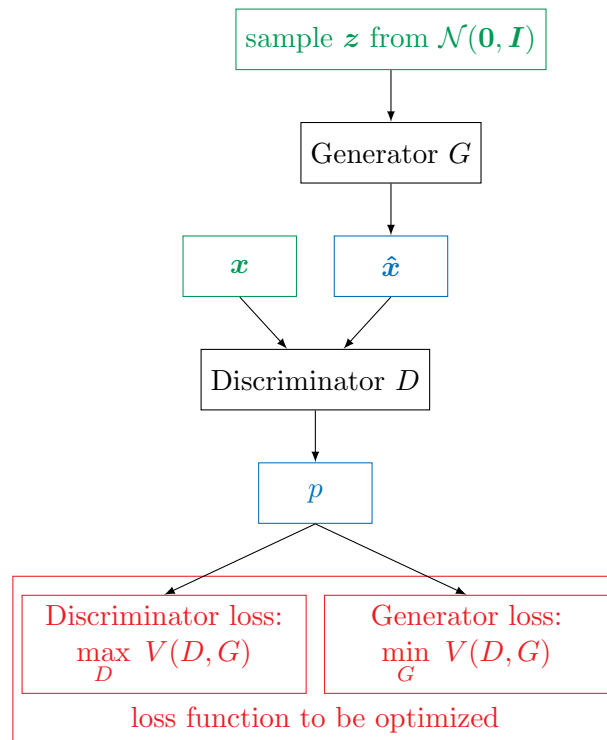


Figure 3.8 – Scheme of a Generative Adversarial Network architecture. In green the inputs, in blue the outputs and in red the terms of the loss function used to compute the gradient descent.

An other powerful alternative is the Generative Adversarial Network (GAN) framework, first introduced in Goodfellow et al. [136]. This framework uses two models that are trained jointly: a discriminator D , and a generator G . The discriminator has to determine if a given sample comes from the generator or from the dataset. The generator has to generate new samples that deceive the discriminator. The idea is that a distribution generating samples able to deceive the discriminator should be close to the distribution at the origin of the dataset. However, approximating the dataset distribution is an implicit goal, as opposed to the VAE where it is set explicitly. The GAN general architecture and data flow is displayed in Figure 3.8.

Let $\mathbf{x} \in \mathcal{X}$ be a random variable and $\mathbf{X} = \{\mathbf{x}^{(i)}\}_{i=1}^n$ a data matrix of samples of this variable, and assume that \mathbf{x} is generated from a distribution $P(\mathbf{x})$. The role of the generator is to learn this distribution. For that, it takes as input a random variable \mathbf{z} sampled from a given distribution, for example $\mathcal{N}(\mathbf{0}, \mathbf{I})$, and maps it to the data space, through a function $G(\mathbf{z}, \boldsymbol{\theta}_g)$ parameterized by $\boldsymbol{\theta}_g$. The function G can be defined by a

feedforward neural network. The role of the discriminator is to output the probability p of a given input sample $\mathbf{x}^{(i)}$ to belong to the dataset \mathbf{X} rather than being generated by G , through a function $D(\mathbf{x}, \boldsymbol{\theta}_d)$ parameterized by $\boldsymbol{\theta}_d$, which can be a feedforward neural network. D is trained to maximize its ability to successfully predict the origin of a given sample, and on the contrary, G is trained so that D labels the generated sample as a real sample. It corresponds to a two-players minmax game, formally described with the following objective function $V(D, G)$:

$$\min_G \max_D V(D, G) = E_{\mathbf{x} \sim \mathbf{X}}[\log D(\mathbf{x}, \boldsymbol{\theta}_d)] + E_{\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})}[\log(1 - D(G(\mathbf{z}, \boldsymbol{\theta}_g), \boldsymbol{\theta}_d))] \quad (3.14)$$

The main advantage of the GAN framework is that it is able to represent very sharp distributions. It is an important feature in image generation for example, where a GAN is able to generate extremely detailed images, an ability difficult to reproduce with other techniques. However, one of the drawback of this technique is the complexity and potential instability of the training process.

Indeed, the generator and discriminator are trained alternately, but the synchronization of this training needs to be carefully tuned. If D is trained too much compared to G , G may be unable to produce any sample that deceives D . Conversely, if G is trained too often relatively to D , it may learn to generate only a few samples $\mathbf{x}^{(i)}$ for all possible values that can be sampled from \mathbf{z} in order to maximize its ability to deceive D .

Moreover, the generator may learn to generate adversarial examples to deceive the discriminator, thus making the learning inefficient [136]. Adversarial examples [137] are examples almost identical to an example present in the database from a human eye, but not associated to the corresponding class by the neural network, or on the contrary, examples associated by the neural network with a very high confidence to a given class, whereas having nothing in common with this class from a human eye.

It is worth noting that a lot of variants of VAE and GAN have been developed, as well as some hybrid methods that combine elements from both architectures [131].

Transformer

An other generative model that receives a lot of interest recently is the transformer framework [138]. This framework is designed to predict sequences of elements. It relies on the Attention mechanism [139]. It has an encoder-decoder structure, with the encoder learning to map a sequence of inputs $(\mathbf{x}_1, \dots, \mathbf{x}_n)$ to a sequence of latent variables $(\mathbf{z}_1, \dots, \mathbf{z}_n)$. In turn, the decoder learns to map a sequence of latent representation to a sequence of outputs $(\mathbf{y}_1, \dots, \mathbf{y}_n)$. This process is performed step by step, one element at a time. The model is also auto-regressive, that is the generated outputs at the previous step is considered as an input for the current step. This network was initially developed for natural language processing, and more specifically translation tasks. It has been extended to image generation, the Attention mechanism allowing to replace convolutional layers [140]. However, to apply this framework, the problem needs to be formulated as a sequence generation problem, which is not always adapted.

3.2.3 Chosen Approach

To extract the grasp space structure and generate new grasps, both dimensionality reduction and generative abilities are required. The dimensionality reduction presented previously are very powerful, but are unable to generate new data, thus they are not suitable for our use case.

Regarding the generative models, the transformer framework is not adapted: the grasp generation problem cannot be formulated easily as a sequence generation problem. GAN is a very powerful tool, but its training instability issues is a drawback that needs to be taken into account. It is able to perform dimensionality reduction, as one can choose the dimension of the input noise. However, the obtained performances can vary significantly depending on the chosen dimension, and there is no clear guidelines for an optimal choice [141].

In this work, the VAE framework has been chosen. Indeed, It has a simpler training process than GAN, and does not has its instability issues. Moreover, its ability to extract low-dimensional manifolds from the data features fits particularly well with our hypothesis of a grasp space that constitute a submanifold of the gripper configuration space. In the following, a more in depth description of the Variational Auto-Encoder is given.

3.3 Variational Auto-Encoder: Principle and Techniques

Variational Auto-Encoder (VAE) has been first introduced in Kingma and Welling [135]. It resembles to classic autoencoder, but is formulated in a probabilistic framework. Recall that the goal is to have a model able to generate new data samples that resemble original samples present in a dataset \mathbf{X} , from some latent variables. The latent variables are unobserved variables that describe main characteristics of a sample, and from which it can be generated. Thus, a necessary condition for the model to give a correct generation is that for any sample in the dataset \mathbf{X} , there exists at least one combination of latent variables able to generate a new data point very similar to this particular sample. The following formalism is inspired from the ones developed in Kingma and Welling [135] and Doersch [142].

3.3.1 Mathematical Principles

Mathematically, a VAE relies on the concept of marginal likelihood, that is a joint probability in which some parameter variables have been marginalized (integrated).

Decoder

Let $\mathbf{X} = \{\mathbf{x}^{(i)}\}_{i=1}^n$ be a dataset of samples of some random variable vector $\mathbf{x} \in \mathcal{X}$ following a distribution $P(\mathbf{x})$ defined over \mathcal{X} . Let $\mathbf{z} \in \mathcal{Z}$ be the latent variable vector, that follows a distribution $P(\mathbf{z})$ defined over \mathcal{Z} . The VAE needs to build an estimation of the unknown distribution $P(\mathbf{x})$. A good approximation of $P(\mathbf{x})$ implies that the VAE

is able to generate samples having a high $P(\mathbf{x})$ value for any latent variables sampled from $P(\mathbf{z})$. For that, the VAE has to maximise the following equation, called marginal likelihood:

$$P(\mathbf{x}) = \int P(\mathbf{x}|\mathbf{z})P(\mathbf{z}) d\mathbf{z} \quad (3.15)$$

However, this equation as expressed here is intractable for three reasons:

- the conditional distribution $P(\mathbf{x}|\mathbf{z})$ is unknown,
- the latent variable vector \mathbf{z} and its associated distribution $P(\mathbf{z})$ is unknown,
- the integral over \mathbf{z} is complex to compute.

Regarding the conditional distribution $P(\mathbf{x}|\mathbf{z})$, the VAE framework makes the assumption that it can be approximated by the following gaussian distribution, parameterized by $\boldsymbol{\theta} \in \Theta$: $P(\mathbf{x}|\mathbf{z}, \boldsymbol{\theta}) = \mathcal{N}(f(\mathbf{z}, \boldsymbol{\theta}), \beta\mathbf{I})$. f is a deterministic function, and $f: \mathcal{Z} \times \Theta \rightarrow \mathcal{X}$. β is an hyperparameter whose influence is discussed later. Without loss of generality, it can be assumed that f is a neural network, with $\boldsymbol{\theta}$ its weights and bias. **The distribution $P(\mathbf{x}|\mathbf{z}, \boldsymbol{\theta})$ corresponds to the decoder side of a classic autoencoder, as it outputs samples of \mathbf{x} given latent variables \mathbf{z} .** Equation 3.15 becomes:

$$P(\mathbf{x}) = \int P(\mathbf{x}|\mathbf{z}, \boldsymbol{\theta})P(\mathbf{z}) d\mathbf{z} \quad (3.16)$$

The goal is to find a vector $\boldsymbol{\theta}$ that maximizes this equation.

Regarding the distribution of \mathbf{z} , the VAE framework assumes that no specific knowledge is required, and that $P(\mathbf{z}) = \mathcal{N}(\mathbf{0}, \mathbf{I})$. However, with such assumption, it is very unlikely that $P(\mathbf{z})$ corresponds to the real latent variable distribution. The fact is that any distribution can be estimated by passing a normal distribution through the appropriate function [143]. In turn, the universal approximation theorem (Theorem 1) states that such function can be represented by a neural network. Thus, the function $f(\mathbf{z}, \boldsymbol{\theta})$ has two goals:

- map the sampled \mathbf{z} to the corresponding “true” latent variables,
- map the latent variables to the corresponding \mathbf{x} value.

The last difficulty is the integral over \mathbf{z} . This is taken care of by the encoder part of the VAE.

Encoder

Regarding the integral over \mathbf{z} , $P(\mathbf{x})$ also corresponds to an expectation and could be estimated by sampling several vectors $\mathbf{z}^{(i)}$ with i from 1 to n , with $P(\mathbf{x}) \approx \frac{1}{n} \sum_i P(\mathbf{x}|\mathbf{z}^{(i)}, \boldsymbol{\theta})$. However, it would require a significant number of samples to find $\boldsymbol{\theta}$ that maximize $P(\mathbf{x})$, as for most \mathbf{z} , $P(\mathbf{x}|\mathbf{z}, \boldsymbol{\theta})$ will be zero or very close to zero. The trick used in the VAE framework to solve this is to only sample \mathbf{z} that are likely to produce a valid \mathbf{x} , so that

$P(\mathbf{x})$ can be estimated and maximized faster. This requires the knowledge of the a priori unknown distribution $P(\mathbf{z}|\mathbf{x}, \boldsymbol{\theta})$. It is unknown as it represents the distribution of \mathbf{z} values that are able to generate some given \mathbf{x} values for a given function $f(\mathbf{z}, \boldsymbol{\theta})$.

To approximate this unknown distribution, a new distribution $Q(\mathbf{z}|\mathbf{x}, \boldsymbol{\phi})$ parameterized by $\boldsymbol{\phi}$ is required, that takes as input a sample of \mathbf{x} and output a distribution over \mathbf{z} values having high probability to generate the given sample of \mathbf{x} . It is assumed that $Q(\mathbf{z}|\mathbf{x}, \boldsymbol{\phi}) = \mathcal{N}(\mu(\mathbf{x}, \boldsymbol{\phi}_1), \text{diag}[\Sigma(\mathbf{x}, \boldsymbol{\phi}_2)])$ with μ and Σ deterministic functions such as $\boldsymbol{\phi}_1 \cup \boldsymbol{\phi}_2 = \boldsymbol{\phi}$. As f , it can be assumed that they are defined by neural networks. **The distribution $Q(\mathbf{z}|\mathbf{x}, \boldsymbol{\phi})$ is analog to the encoder of a classic autoencoder, as it encodes a sample of \mathbf{x} in the latent variables \mathbf{z} .** The goal is to find a parameter vector $\boldsymbol{\phi}$ that makes the two distributions $Q(\mathbf{z}|\mathbf{x}, \boldsymbol{\phi})$ and $P(\mathbf{z}|\mathbf{x}, \boldsymbol{\theta})$ as close as possible.

The considered graphical model representing the VAE is displayed in Figure 3.9.

Objective Function

With these assumptions, a lower bound of the marginal likelihood can then be estimated as follows [135, 142]:

$$\underbrace{\log P(\mathbf{x}) - \mathcal{D}[Q(\mathbf{z}|\mathbf{x}, \boldsymbol{\phi})\|P(\mathbf{z}|\mathbf{x}, \boldsymbol{\theta})]}_{\text{lower bound of the marginal likelihood}} = \mathbb{E}_{\mathbf{z} \sim Q}[\log P(\mathbf{x}|\mathbf{z}, \boldsymbol{\theta})] - \mathcal{D}[Q(\mathbf{z}|\mathbf{x}, \boldsymbol{\phi})\|P(\mathbf{z})] \quad (3.17)$$



Definition 14 (Kullback-Leibler divergence). The Kullback-Leibler divergence (or KL divergence) $\mathcal{D}[P_1\|P_2]$ is a statistical distance: a measure of how one probability distribution P_2 is different from a second, reference probability distribution P_1 . It was firstly introduced by Kullback and Leibler [144]. A simple interpretation of the divergence of P_1 from P_2 is the expected excess surprise from using P_2 as a model when the actual distribution is P_1 .

The KL divergence being non-negative, the left term of the equation is a lower bound of the marginal likelihood $\log P(\mathbf{x})$. Recall that the goal is to maximize the marginal likelihood. Thus, an efficient way for that is to maximize its lower bound. To maximize it, the KL divergence between $Q(\mathbf{z}|\mathbf{x}, \boldsymbol{\phi})$ and $P(\mathbf{z}|\mathbf{x}, \boldsymbol{\theta})$ needs to be minimized, but the distribution $P(\mathbf{z}|\mathbf{x}, \boldsymbol{\theta})$ is intractable. Instead, the lower bound can be maximized by maximizing the right hand side of the equation, which can be optimized through gradient descent.

As the gradient descent goes along, the first term of the right hand side makes $Q(\mathbf{z}|\mathbf{x}, \boldsymbol{\phi})$ more likely to produce \mathbf{z} samples that maximize $P(\mathbf{x}|\mathbf{z}, \boldsymbol{\theta})$. This implicitly minimizes the KL divergence between $Q(\mathbf{z}|\mathbf{x}, \boldsymbol{\phi})$ and $P(\mathbf{z}|\mathbf{x}, \boldsymbol{\theta})$, as it gradually makes $Q(\mathbf{z}|\mathbf{x}, \boldsymbol{\phi})$ a better approximation of $P(\mathbf{z}|\mathbf{x}, \boldsymbol{\theta})$. Thus optimizing the right hand side of the equation will become equivalent to optimizing $\log P(\mathbf{x})$.

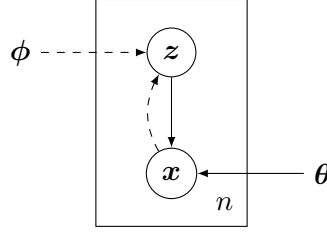


Figure 3.9 – Directed graphical model of a VAE [135]. Solid lines describe the generative model $P(\mathbf{x}|\mathbf{z}, \boldsymbol{\theta})P(\mathbf{z})$, and dashed lines represent the approximation $Q(\mathbf{z}|\mathbf{x}, \boldsymbol{\phi})$ of the intractable distribution $P(\mathbf{z}|\mathbf{x}, \boldsymbol{\theta})$. The parameter $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$ are learned simultaneously by sampling in the dataset \mathbf{X} . One can sample n times a \mathbf{x} or \mathbf{z} with fixed $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$.

3.3.2 Training and Generating Samples

Gradient Descent Optimization and Reparameterization Trick

The training process of a VAE together with the reparameterization trick is summarized in Figure 3.10.

To perform the gradient descent algorithm during the training phase, Equation 3.17 is evaluated for several \mathbf{x} values sampled from the dataset \mathbf{X} , and the equation becomes:

$$\mathbb{E}_{\mathbf{x} \sim \mathbf{X}} [\log P(\mathbf{x}) - \mathcal{D}[Q(\mathbf{z}|\mathbf{x}, \boldsymbol{\phi}) \| P(\mathbf{z}|\mathbf{x}, \boldsymbol{\theta})]] = \underbrace{\mathbb{E}_{\mathbf{x} \sim \mathbf{X}} [\mathbb{E}_{\mathbf{z} \sim Q} [\log P(\mathbf{x}|\mathbf{z}, \boldsymbol{\theta})]]}_{\text{reconstruction loss}} - \underbrace{\mathcal{D}[Q(\mathbf{z}|\mathbf{x}, \boldsymbol{\phi}) \| P(\mathbf{z})]}_{\text{KL divergence loss}} \quad (3.18)$$

Reparameterization Trick Computing $\mathbb{E}_{\mathbf{z} \sim Q} [\log P(\mathbf{x}|\mathbf{z}, \boldsymbol{\theta})]$ requires to sample \mathbf{z} values from the distribution $Q(\mathbf{z}|\mathbf{x}, \boldsymbol{\phi}) = \mathcal{N}(\mu(\mathbf{x}, \boldsymbol{\phi}_1), \text{diag}[\Sigma(\mathbf{x}, \boldsymbol{\phi}_2)])$. This sampling operation does not allow the gradient back-propagation. The reparameterization trick, proposed in Kingma and Welling [135], allows to solve this issue by replacing this sampling operation by the following:

$$\mathbf{z} = \mu(\mathbf{x}, \boldsymbol{\phi}_1) + \Sigma(\mathbf{x}, \boldsymbol{\phi}_2)^{\frac{1}{2}} \odot \boldsymbol{\varepsilon}, \quad \boldsymbol{\varepsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}) \quad (3.19)$$

with \odot a componentwise multiplication. After applying the reparameterization trick, Equation 3.18 becomes:

$$\mathbb{E}_{\mathbf{x} \sim \mathbf{X}} [\log P(\mathbf{x}) - \mathcal{D}[Q(\mathbf{z}|\mathbf{x}, \boldsymbol{\phi}) \| P(\mathbf{z}|\mathbf{x}, \boldsymbol{\theta})]] = \mathbb{E}_{\mathbf{x} \sim \mathbf{X}} [\mathbb{E}_{\boldsymbol{\varepsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})} [\log P(\mathbf{x}|\mathbf{z} = \mu(\mathbf{x}, \boldsymbol{\phi}_1) + \Sigma(\mathbf{x}, \boldsymbol{\phi}_2)^{\frac{1}{2}} \odot \boldsymbol{\varepsilon}, \boldsymbol{\theta})] - \mathcal{D}[Q(\mathbf{z}|\mathbf{x}, \boldsymbol{\phi}) \| P(\mathbf{z})]] \quad (3.20)$$

With some given samples \mathbf{x} and $\boldsymbol{\varepsilon}$, this equation is deterministic and the gradient can be back-propagated.

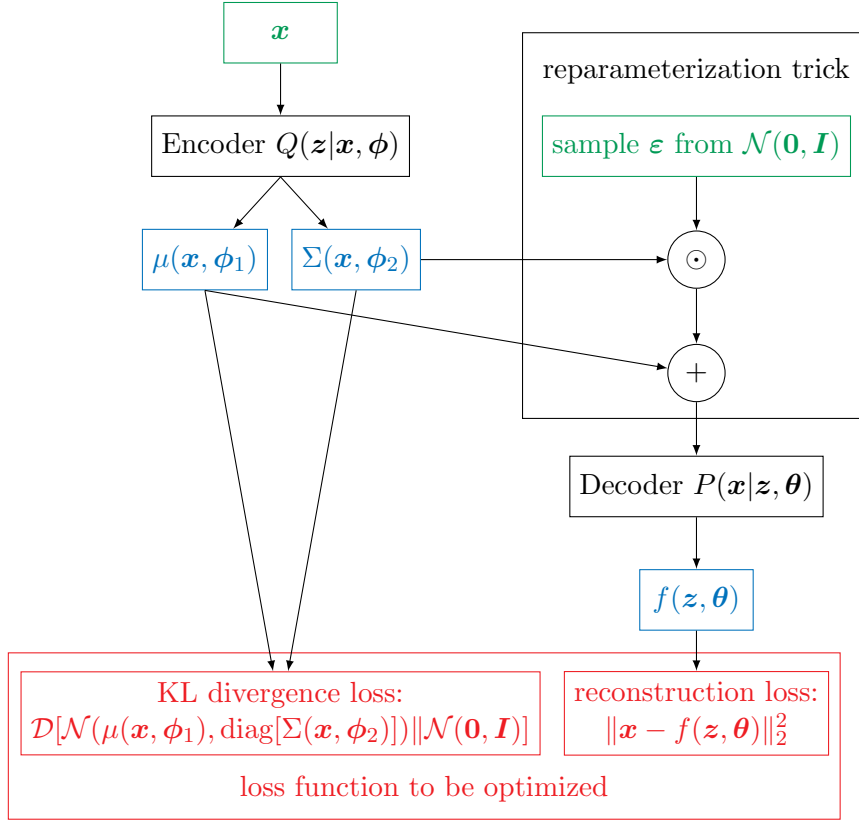


Figure 3.10 – Scheme of the training process of the VAE, with the reparameterization trick. In green the encoder and decoder inputs, in blue their outputs, and in red the terms of the loss function used for the gradient descent.

KL Divergence Loss Component With the chosen hypothesis of a Gaussian distribution for the distribution $Q(z|x, \phi)$ and $P(z)$, the KL divergence loss component on the right side of Equation 3.18 admits a simple form and can be computed as follows:

$$-\mathcal{D}[Q(z|x, \phi) \parallel P(z)] = -\mathcal{D}[\mathcal{N}(\mu(x, \phi_1), \text{diag}[\Sigma(x, \phi_2)]) \parallel \mathcal{N}(\mathbf{0}, \mathbf{I})] \quad (3.21)$$

$$= \frac{1}{2} \sum_{j=1}^{\dim \mathcal{Z}} (\log(\Sigma_j) - (\mu_j)^2 - \Sigma_j) \quad (3.22)$$

Reconstruction Loss Component The term $\log P(x|z, \theta)$ of Equation 3.18 can be seen as a reconstruction loss. Maximizing it is equivalent to minimizing the mean squared error between the input data and the estimated data, that is minimizing $\|x - f(z, \theta)\|_2^2$ [118]. In that sense, it is similar to the reconstruction loss of a classic autoencoder.

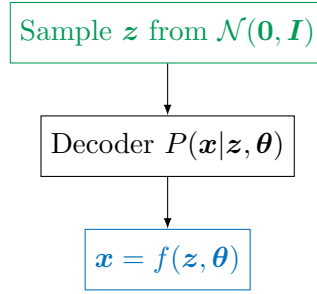


Figure 3.11 – Scheme of a VAE after the training, when used to generate new data: only the decoder part is used, and the latent variables are sampled from $P(\mathbf{z}) = \mathcal{N}(\mathbf{0}, \mathbf{I})$. In green the decoder input, and in blue its output.

Generating Samples

Once the training is completed, ideally the VAE has captured the distribution $P(\mathbf{x})$. New samples of \mathbf{x} can be generated by sampling \mathbf{z} vectors from $\mathcal{N}(\mathbf{0}, \mathbf{I})$, and using them as input to the decoder. This is summarized in Figure 3.11.

3.3.3 VAE Variants

β -VAE

An hyperparameter has been introduced in the VAE framework, called β in Higgins et al. [145]. Depending on authors, this parameter is introduced as a coefficient on the denominator when computing $\log P(\mathbf{x}|\mathbf{z}, \boldsymbol{\theta})$ [142] or on the numerator of the KL divergence term [145] on the right side of Equation 3.18. Both point of views have equivalent interpretation. This parameter can be seen as a regularization parameter, allowing to change the weight granted to the KL divergence constraint relative to the reconstruction of the correct sample of \mathbf{x} . Indeed, the KL divergence term tends to reduce the amount of information contained in $Q(\mathbf{z}|\mathbf{x}, \boldsymbol{\phi})$ to make it as close as possible to the non-informative $P(\mathbf{z})$.

- Putting more weight on this constraint reduces the encoder ability to output a \mathbf{z} that is likely to produce the required \mathbf{x} , and thus reduces the ability of the decoder to reconstruct correctly \mathbf{x} given the misleading \mathbf{z} . However, such setup forces the encoder and decoder to extract the most important features of the samples, and produces a latent space with each latent variable storing one of these features, each of them being decorrelated from the others [145, 146].
- If the weight on the KL divergence constraint is reduced, the encoder will be able to produce very informative \mathbf{z} samples, thus allowing the decoder to reconstruct more accurately the \mathbf{x} values. In this case, the main drawback is that the decoder will rely more on its encoder to produce a meaningful reconstruction. The distribution $P(\mathbf{z})$ produced by the encoder may move away from the standard normal assumption

during training. However, at test time, \mathbf{z} values are sampled from a standard normal distribution. The VAE may fail to produce samples consistent with $P(\mathbf{x})$ when its input is a \mathbf{z} sample from $P(\mathbf{z})$ instead of $Q(\mathbf{z}|\mathbf{x}, \phi)$.

It is worth noting that the model learnt by a VAE with $\beta = 0$ is equivalent to a classic autoencoder, as the encoder will learn early in the training to output only punctual distributions, as it is the best way to maximize the quality of the reconstruction. Thus, a trade-off needs to be found between the reconstruction abilities, and the regularity of the latent space. It is this latent space regularity that makes VAE good at learning low-dimensional manifolds.

Conditional VAE

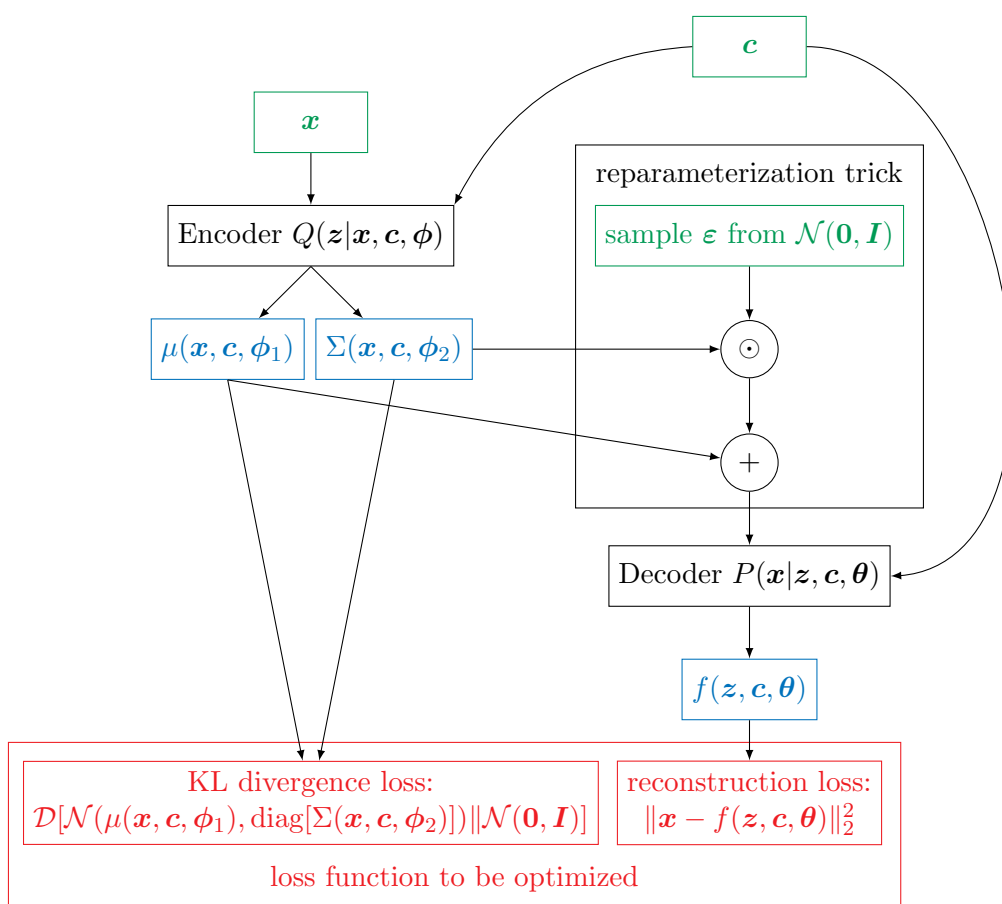


Figure 3.12 – Scheme of the training process of the CVAE, with the reparameterization trick. In green the encoder and decoder inputs, in blue their outputs, and in red the terms of the loss function used for the gradient descent.

A variant of the VAE is the Conditional Variational Auto-Encoder (CVAE) [147].

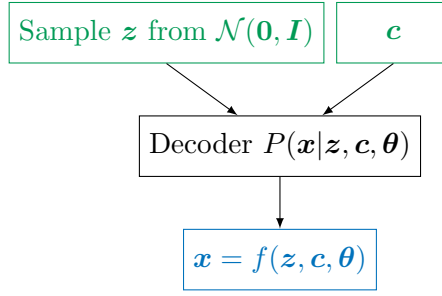


Figure 3.13 – Scheme of a CVAE after the training, when used to generate new data: only the decoder part is used, and the latent variables are sampled from $P(\mathbf{z}) = \mathcal{N}(\mathbf{0}, \mathbf{I})$. In green the decoder input, and in blue its output.

The main idea is to condition output generated by the VAE on a supplementary input \mathbf{c} . In this framework, the encoder and decoder distributions are conditioned by \mathbf{c} and become $Q(\mathbf{z}|\mathbf{x}, \mathbf{c}, \phi)$ and $P(\mathbf{x}|\mathbf{z}, \mathbf{c}, \theta)$. Equation 3.17 can be rewritten as:

$$\log P(\mathbf{x}|\mathbf{c}) - \mathcal{D}[Q(\mathbf{z}|\mathbf{x}, \mathbf{c}, \phi) \| P(\mathbf{z}|\mathbf{x}, \mathbf{c}, \theta)] = E_{\mathbf{z} \sim Q}[\log P(\mathbf{x}|\mathbf{z}, \mathbf{c}, \theta)] - \mathcal{D}[Q(\mathbf{z}|\mathbf{x}, \mathbf{c}, \phi) \| P(\mathbf{z}|\mathbf{c})] \quad (3.23)$$

The training process of a CVAE is displayed in Figure 3.12. It is identical to the classic VAE one, the information of the condition \mathbf{c} simply needs to be added to the encoder and decoder. At testing time, when generating new samples \mathbf{x} , the principle is the same as for a classic VAE, a vector \mathbf{z} sampled from $\mathcal{N}(\mathbf{0}, \mathbf{I})$ is used as input, but in addition a conditioning data \mathbf{c} needs to be provided. This is summarized in Figure 3.13.

Now that the tool used for grasp generation is presented, the principle of the proposed method is described in the following.

3.4 Variational Auto-Encoders for Grasp Space Exploration

The method proposed in this work takes advantage from human experience to guide the grasp space exploration. Indeed, it is easy for a human to find gripper configurations that are likely to grasp a given object, that is configurations belonging to the grasp space. However, those primitive grasps do not necessarily have optimal quality metric values. Actually, it is difficult for a human to assess a priori the relative and absolute quality of grasp configurations. The aim of the grasp space exploration is to search for grasps by testing gripper configurations close to the ones proposed by the human and to predict the quality of each explored grasp. This is done by interpolating and extrapolating new grasps using VAE. Exploring the grasp space allows to build a collection of grasps with various quality values, including grasps with higher quality than the primitive ones if such grasps exist.

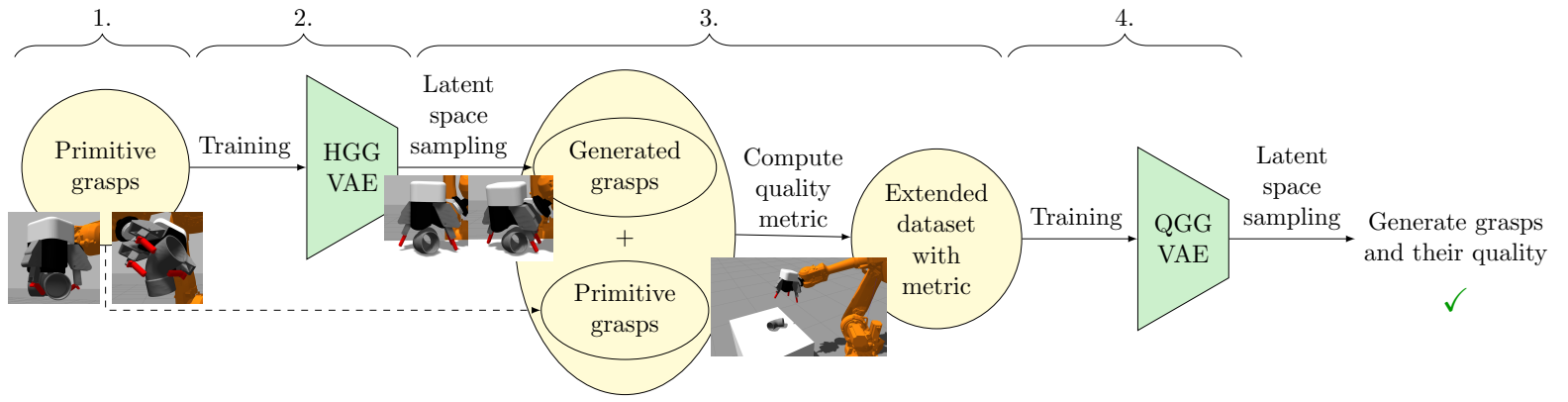


Figure 3.14 – Scheme of the presented workflow.

Exploring the grasp space taking inspiration from human-provided grasps allows to massively reduce the configuration space to be explored compared to exploration methods based on random sampling, while retaining sufficient exploration potential. Indeed, classic methods based on random sampling require to cover as much gripper configuration space as possible to find possible grasps, and often produce a high number of failed grasps. With the proposed method, there is no need to cover the whole gripper configuration space to find suitable grasps, as the exploration is already focused on configurations with a high success probability from human expertise. Predicting a grasp quality allows to improve even further the success rate of generated grasps.

The general workflow used to achieve this is summarized on Figure 3.14 and is decomposed as follows:

1. the constitution of a primitive grasp dataset
2. the training of a Human-initiated Grasp Generator Variational Autoencoder (HGG)
3. a dataset extension & grasp quality estimation phase
4. the training of a Quality-oriented Grasp Generator Variational Autoencoder (QGG)

Each of the above steps is detailed below.

3.4.1 Primitives Dataset

To leverage the human ability to find gripper configurations belonging to the grasp space, an object dependent primitive grasp dataset is built. A primitive grasp is a handcrafted gripper configuration, with its pose and gripper internal configuration human-chosen so that it is collision free and likely to grasp the object from human expertise. In our case, these primitive grasps are specified thanks to a simulation: the operator moves the simulated gripper around an object and register the desired pose. This procedure could also be applied on a real setup, through physical demonstration using a cobot for example.

For a given object, several grasp types or categories can be identified from human experience. For example, for a cylinder, two possible grasp types are grasps with the palm along the cylinder side, and grasps with the palm along one of the cylinder base. This is illustrated in Figure 3.15. For each of these grasp types, several variations need to be provided so that the VAE can extract the underlying manifold structure.

The dataset stores the $d_{conf} + 1$ parameters describing the gripper configuration (see subsection 3.1.2) of each primitive grasp along with the four parameters of the tabletop plane Cartesian equation, both in object frame F_{obj} . Knowing the stable position of the object is a critical information to avoid collisions. Some grasps may collide with the table in a given stable position, while being suitable for an other one.

Expressing the grasp configuration in the object frame is still useful as it allows an invariance to a position change and to a rotation around a vertical axis.

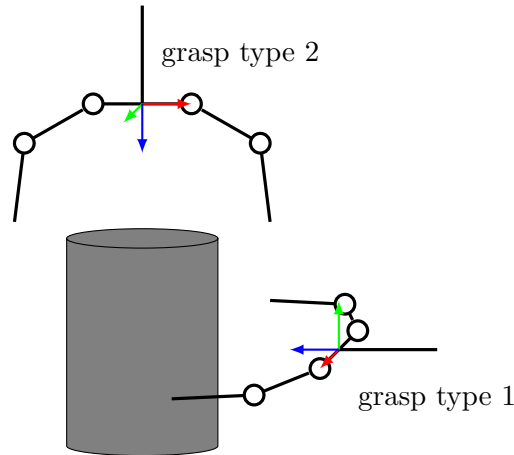


Figure 3.15 – Example of two grasp configurations belonging to two different grasp types on a cylinder, for a two fingered gripper with two phalanges.

3.4.2 Human-initiated Grasp Generator (HGG)

The goal of the HGG is to infer the correlations existing between the parameters of different grasp primitives to learn a model of the grasp space. Such correlations exist, as primitive grasps are in the grasp space, and the grasp space is a subset of the gripper configuration space. The HGG is able to use those correlations to map the grasp space in its latent space. This allows the relevant grasps to be densely represented in the latent space, contrary to the gripper configuration space, in which they are present in a very sparse way.

Concretely, the HGG is a CVAE, that has to predict the distribution at the origin of the primitive grasps and by extension of the grasp space, conditioned on the tabletop plane Cartesian equation. Its training dataflow is shown in Figure 3.16. It is trained for a given object on the grasp primitive dataset. This model can then be used to generate efficiently new configurations that have a high probability to be in the grasp space. It is worth noting that the choice of representing orientation with quaternions allows to ease the generation process, as a simple linear interpolation between two quaternions can give a consistent new orientation, which is not the case with euler angles or rotation matrix representation. Thus, the network will need less capacity to interpolate and extrapolate new orientations.

3.4.3 Dataset Extension & Grasp Quality Computation

Sampling in the latent space of the HGG allows to explore the grasp space in a efficient way. Indeed, the HGG takes into account the correlations existing between the parameters of the primitive grasp configurations, and thus is able to extrapolate and interpolate new grasps that take inspiration from human grasp strategies.

These generated grasps are tested in simulation along with primitive ones to check

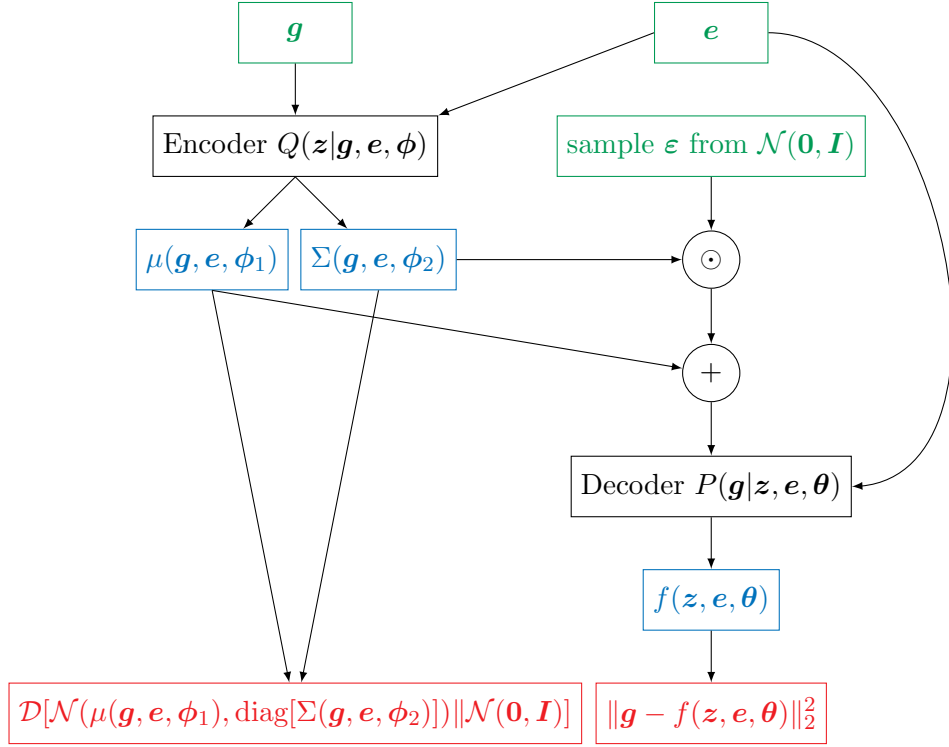


Figure 3.16 – Scheme of the training workflow of the HGG, with g the grasp configuration, and e the tabletop Cartesian equation as defined in subsection 3.1.2. In green the inputs, in blue the outputs and in red the loss functions used during the training.

their success. A configuration is successful if the following conditions are met:

- it does not collide with the table;
- it successfully lifts the object from the table;
- its Q_{MSV} value is greater than 0.

For each successful configuration, the computed Q_{MSV} quality value is registered. For failed configurations, a null value is registered as quality value.

This allows to extend the primitive dataset by exploring extensively the grasp space. Thus, a collection of grasps with various quality values can be constituted, and if better grasps than the primitive ones exist, they can be discovered and stored for the following step.

3.4.4 Quality-oriented Grasp Generator (QGG)

The goal of the QGG is to reliably generate grasps with their corresponding grasp quality. As the HGG, it is a CVAE, that has to predict the distribution at the origin of the grasp space, conditioned on the tabletop plane Cartesian equation. The decoder part is slightly

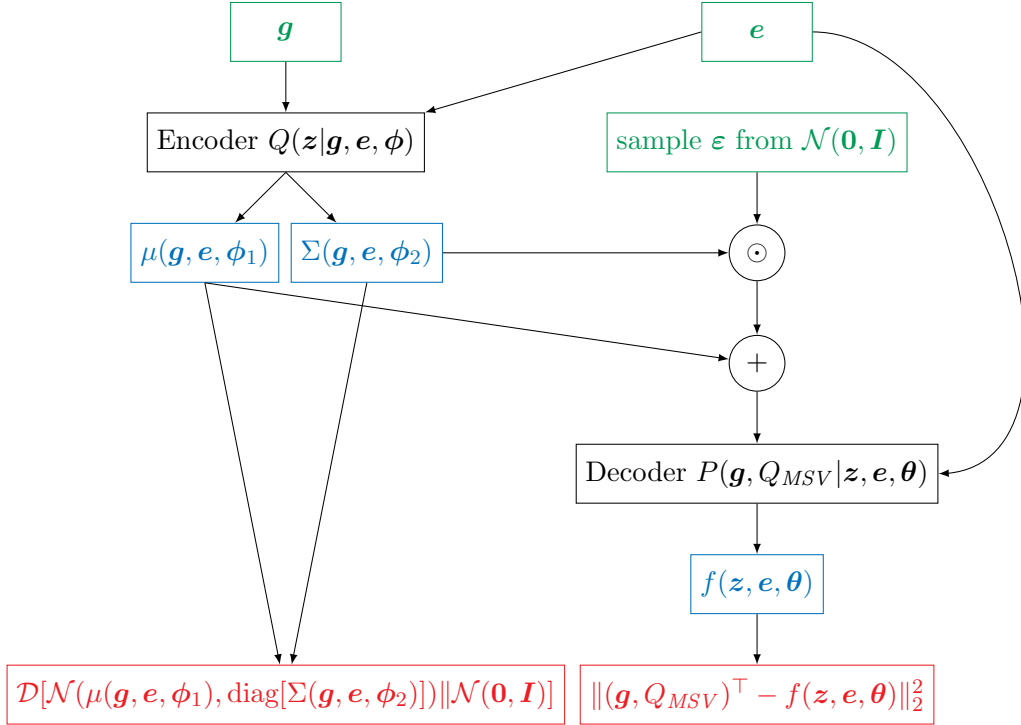


Figure 3.17 – Scheme of the training workflow of the QGG, with g the grasp configuration and e the tabletop Cartesian equation as defined in subsection 3.1.2, and Q_{MSV} the grasp quality. In green the inputs, in blue the outputs, and in red the loss functions used during the training.

modified as it also has to perform a regression on the quality Q_{MSV} value. A scheme of the training architecture is displayed in Figure 3.17.

The QGG is trained for a given object on the extended set formed by merging the primitive grasp set with the generated grasp set (both successful and failed). Learning failed grasps together with successful ones allows to identify areas of the configuration space that are close to the grasp space but unable to produce a successful grasp, to reduce the risk of predicting a high quality for such configurations. For example, some configurations can lead to a failed grasp because of collisions or object movements during the finger closing phase. These configurations are labeled with a null grasp quality. The QGG can then learn to predict a null or very low grasp quality associated with these bad configurations. Moreover, the dataset extension allows to represent more accurately and more reliably the grasp space.

The QGG can be used to explore the grasp space in an even more efficient way than the HGG. Indeed, the grasp quality prediction allows to filter the generated grasp based on their expected quality. Grasps having a high quality and a high probability of success can be generated simply by selecting the ones with a quality prediction above a given threshold.

3.5 Conclusion

This chapter has introduced a method for an efficient grasp space exploration adapted to a pluri-digital and underactuated gripper. This method aims at extracting the structure of the grasp space, from a set of human-provided grasp primitives, in order to generate efficiently realistic grasps. For that, it uses the VAE framework.

First, the considered hypotheses along with the required inputs and outputs have been presented. Then, a description of the state of the art regarding dimensionality reduction and generative model have been made to highlight the various tools available for these issues. Next, the theoretical knowledge regarding VAE has been explained. Finally, The different steps of the proposed method have been detailed.

The aim of the next chapter is to present a concrete implementation of this method, assess its performances, both in simulation and on a real setup, and compare it with other classic sampling-based methods.

Chapter 4

Method Qualification

In this chapter, the grasp space exploration method proposed in chapter 3 is detailed, implemented and qualified. The described workflow is applied on several objects, using an underactuated gripper having three two-phalanx fingers designed at CEA List. This setup is described, together with the chosen objects, corresponding primitive grasps, and VAEs architecture. An in depth analysis of the produced latent space is proposed, and the VAEs hyperparameter tuning is discussed. Finally, the proposed method is compared with a grasp space exploration method based on random sampling, and its performances are assessed, both in simulation and in a real robotic demonstrator.

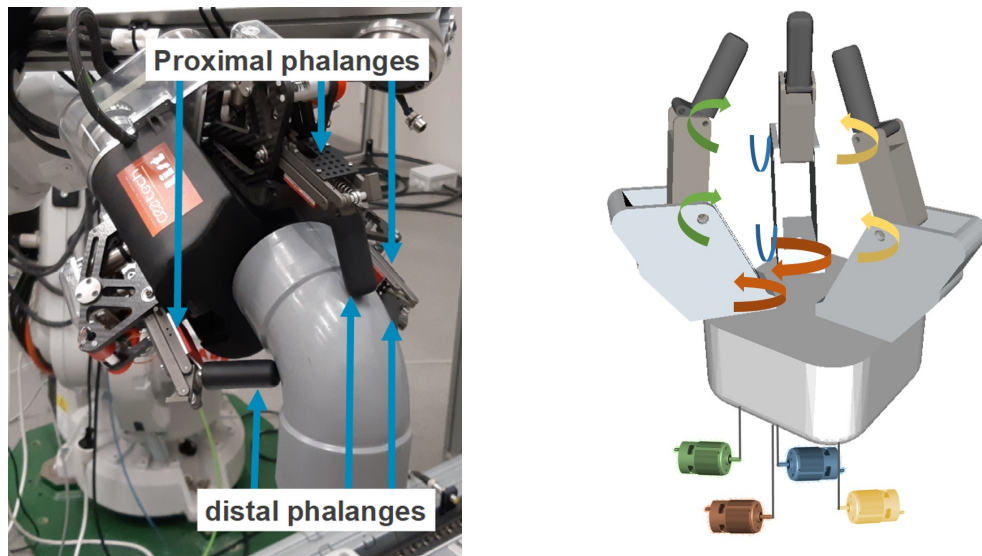
Contents

4.1	Considered Setup, Objects and Primitives Grasps	122
4.1.1	Considered Gripper and Robot	122
4.1.2	Simulation Setup	124
4.1.3	Objects and Corresponding Primitives Grasps	125
4.1.4	HGG & QGG Networks Architecture and Training	127
4.2	Hyperparameters Tuning and Latent Space Analysis	129
4.2.1	HGG Latent Space: Illustration in the Case of Two Latent Variables	129
4.2.2	HGG Tuning to Model Efficiently the Grasp Space	131
4.3	Experiments	137
4.3.1	Comparison with Classic Sampling-Based Grasp Space Exploration Methods	137
4.3.2	Simulation Trials	140
4.3.3	Physical Trials	142
4.4	Conclusion	144

4.1 Considered Setup, Objects and Primitives Grasps

The experimental setup on which the method described in section 3.4 is applied is presented in the following. The objects chosen to test the method are also presented, along with the selected primitive grasps. Then, some details are given on the training and architecture of the HGG and QGG.

4.1.1 Considered Gripper and Robot



(a) Picture of the underactuated gripper when grasping an object. (b) Scheme of the actuator-to-joint mapping.

Figure 4.1 – Picture and actuation scheme of the CEA three-fingered underactuated gripper used in this work.

The two-phalanx three-fingered gripper considered in this work was designed at CEA List. It is displayed in Figure 4.1a. It implements an underactuated and adaptive behavior based on differential mechanisms as described in subsection 1.2.3 that allows its natural adaptation to the object geometry, thus increasing the robustness of the grasp, by forming a power grasps.

This gripper has two joints on each finger and one actuator per finger to control both joints. A fourth actuator allows to control in a coupled way the spread angle θ between two fingers, allowing an abduction-adduction motion (see Figure 4.1b for the actuator-to-joint mapping). In each finger, the underactuation is created by a planar four-bar linkage together with an integrated passive elastic spring, located in the proximal phalanx. When the mechanism is at rest, the finger is out straight. The distal phalanx starts moving when the effort applied by the object on the finger is above a given force threshold, determined by the passive elastic spring stiffness. More mathematical details

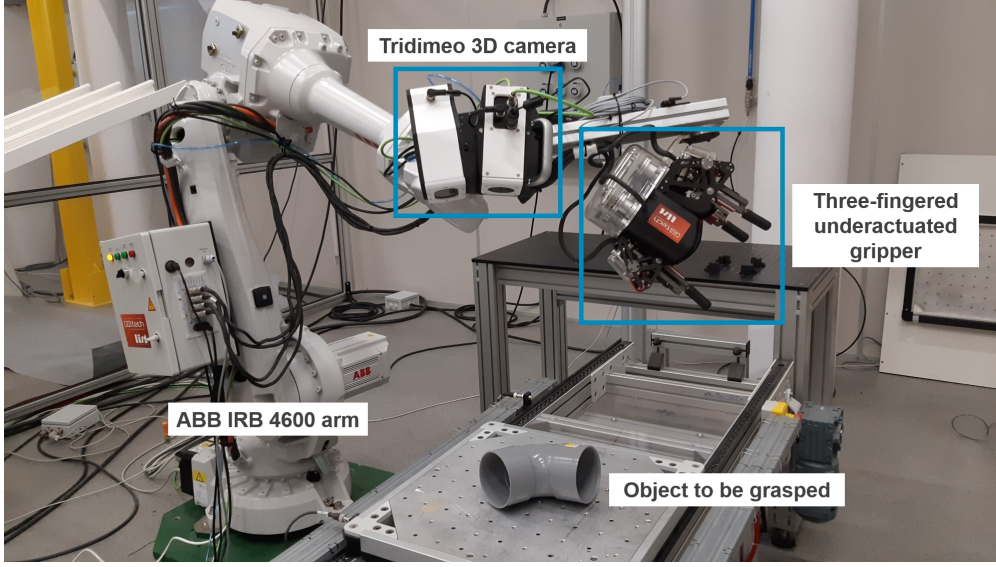


Figure 4.2 – Gripper and camera mounted on an ABB IRB 4600 industrial robot arm.

are available in Appendix A.

Using notations previously introduced in subsection 3.1.1 (see Figure 3.2), this gripper has $n_f = 3$ fingers, with $n_p = 2$ phalanges for each finger. Fingers 1 and 2 have $k_1 = k_2 = 1$ actuated degree of freedom allowing finger repositioning relative to the palm, with a mechanical coupling such as $q_1^1 = -q_1^2 = \theta$. Thus, the gripper internal configuration space has one dimension, and the dimension of the configuration space is then $d_{conf} = 7$. For this gripper, a grasp configuration is defined by the following eight parameters:

- the pose of the gripper frame F_{grip} in the object frame F_{obj} , as exposed in subsection 3.1.2, with the orientation in quaternion convention,

$$(x, y, z, q_x, q_y, q_z, q_w) \in \text{SE}(3) \quad (4.1)$$

- The gripper internal configuration, through the spread angle θ , as shown on Figure 4.3,

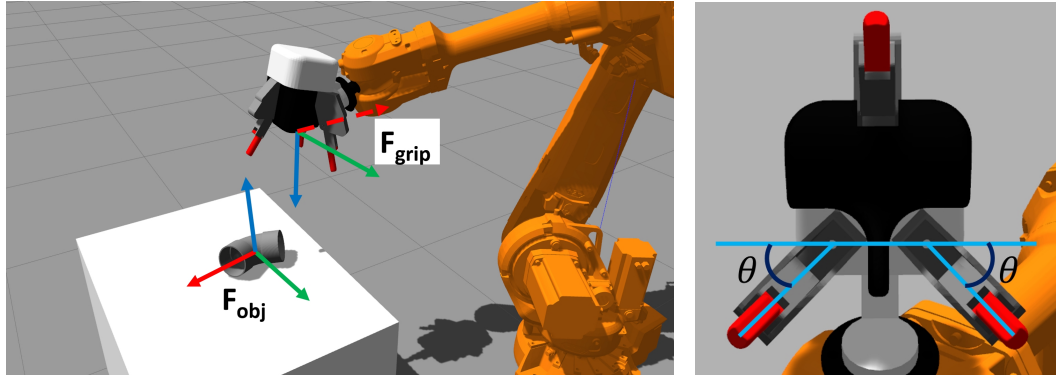
$$\theta \in \left[0, \frac{\pi}{2}\right] \quad (4.2)$$

This gripper is mounted as end effector of a six degrees of freedom industrial robot arm (ABB IRB 4600 model) together with a 3D camera from Tridimeo [148], as shown on Figure 4.2. Details on the global control architecture used are given in Appendix A.

As stated in subsection 3.1.1, our method assumes that the object model is known, as well as its pose. The Tridimeo camera is associated with a software that tries to fit the known object model to a point cloud captured by the camera, based on algorithms given in Mayran de Chamisso et al. [149, 150]. When this fitting is successful, it allows to retrieve the pose of the known object frame in the scene, and express it in the camera

frame. Then, the camera being fixed on the robot, it is possible to express the pose of the object in the reference world frame.

4.1.2 Simulation Setup



(a) Pose of the frame F_{grip} used to locate the end-effector relative to the object frame F_{obj} .

(b) Spread angle θ .

Figure 4.3 – Gripper frame F_{grip} relative to the object frame F_{obj} , and spread angle θ .

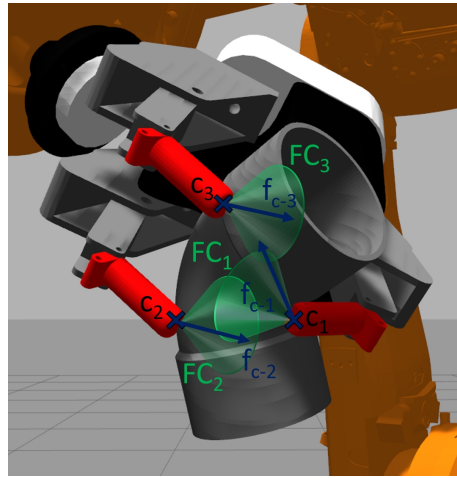


Figure 4.4 – Example of a grasp with four contacts: three contacts with the fingertips c_1 , c_2 , c_3 , and one contact with the palm, not represented as hidden by the object. FC_i is the friction cone of the contact i , and f_{c-i} the contact force for the contact i .

The experimental setup is replicated in simulation using ROS [151] and Gazebo [152]. A picture of this simulated setup is displayed in Figure 4.3.

The underactuation and coupling mechanism are implemented in the Gazebo simulation through a control plugin, which represents low level controllers and converts positions and velocities setpoints into torques applied on the robot arm and gripper joints.

The positions and velocities setpoints for the robot arm joints and gripper actuators are generated by ROS trajectory controllers [153], taking as input the target and current positions. The robot arm joint target positions are determined from the gripper target pose through an inverse kinematics algorithm using MoveIt [154], a motion planning framework integrated with ROS.

The simulation allows to model contacts, in order to compute the grasp quality Q_{MSV} described in Equation 2.83. In the simulation, a point contact with friction model is used. As an example, a grasp involving four contacts is displayed in Figure 4.4: three on the fingers, and one on the palm. The contact positions c and frame orientations (through the directions of the contact normals, that is the axes of the friction cones FC) are retrieved and allow to express the grasp map G , which in turn allows to compute Q_{MSV} .

4.1.3 Objects and Corresponding Primitives Grasps

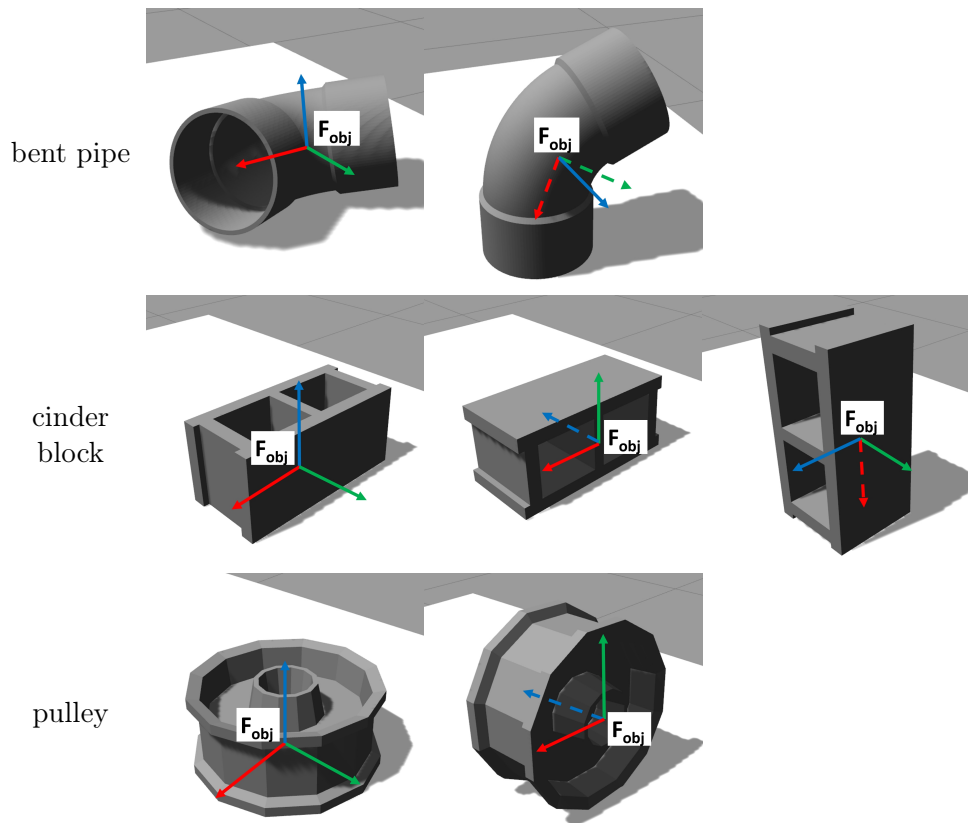


Figure 4.5 – The chosen objects and their frame F_{obj} in their different stable poses.

The method proposed in section 3.4 is applied on three different objects:

- a connector bent pipe

- a pulley
- a small cinder block

Their CAD model (3D meshes) used in the simulation in their different stable poses are visible in Figure 4.5. Those objects have been chosen for their relative complexity and diversity in terms of shapes. Other objects could have been chosen, provided that they have a size, shape and weight compatible with the gripper workspace and payload.

A set of primitive gripper configurations is determined for each of those objects for each of their stable pose. These primitive gripper configurations can be sorted in different grasp types presented in Figure 4.6. Five grasp types can be identified for the bent pipe and cinder block, and three for the pulley. For each of these grasp types, several variants are manually created. For the constitution of this primitive grasp dataset, the spread angle θ is chosen between four discrete values corresponding to main gripper internal layouts: $\theta = 0$, $\theta = \pi/6$, $\theta = \pi/4$, and $\theta = \pi/2$.

The following number of primitive grasps are gathered for each object:

- bent pipe: 145 samples
- cinder block: 141 samples
- pulley: 118 samples

Around one hour has been required for a human operator to register the primitives for a given object in the simulated environment. It is worth noting that the required

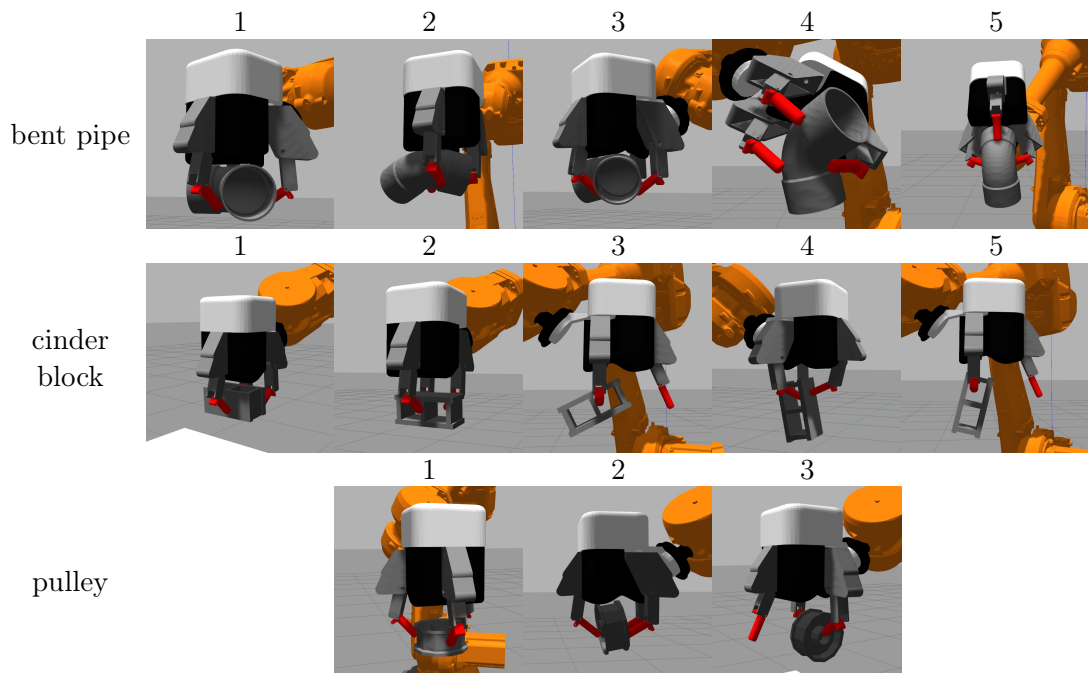


Figure 4.6 – Primitive grasp types for the three chosen objects.

amount of time highly depends on the user interface employed to register the primitives. Here, the interface design is not optimal, and a reduction in the time required can be expected when using an improved interface, or a virtual reality setup for example.

4.1.4 HGG & QGG Networks Architecture and Training

A distinct HGG is trained on the primitive grasp dataset for each object. Its inputs and outputs are shown in Figure 4.7 along with its global architecture. For the gradient descent during the training, a Mean Square Error (MSE) is computed for each gripper parameter. Each of these errors is averaged on each batch of training data. Then, the global loss for each batch is computed as the sum of these averaged errors together with the KL divergence loss. This loss is then used by the RMSprop optimizer implemented in Keras library [155].

To make sure that the quaternion outputs by the decoder is a unit quaternion, a custom activation function is used to normalize the quaternion on the output layer of the decoder.

At first, preliminary tests have been conducted with loss components having a more physical meaning than MSE for the position and orientation reconstruction:

- the mean Euclidean distance between true and reconstructed positions,
- the mean angle α between true and reconstructed orientations, computed by:

$$\alpha = \cos^{-1}(2(\mathbf{q} \cdot \hat{\mathbf{q}})^2 - 1) \quad (4.3)$$

with \mathbf{q} the true orientation, $\hat{\mathbf{q}}$ the reconstructed orientation, and \cdot the quaternion inner product.

Both metrics also allow to compute the reconstruction performances achieved on the training dataset at the end of the training, in order to make a comparison with training using classic MSE loss. It appears that a training using Euclidean distance loss for the position reaches performances equivalent to a training with classic MSE loss. On the contrary, a training using the mean angle loss for orientation obtains lower performances than a training with MSE loss. The performance gap for the orientation between MSE and mean angle loss may be linked to the fact that a given orientation can be represented by two quaternions, \mathbf{q} and $-\mathbf{q}$, which cannot be discriminated by the mean angle loss. With the mean angle loss, every orientation admit two equivalent representations, whereas there is always only one possible representation with the MSE loss. With the mean angle loss, one of the two representations is easier to learn for a given data point, depending on the network weight initialization, but the easiest representation may differ from one data point to an other. After a training with a mean angle loss, for \mathbf{q}_1 and \mathbf{q}_2 two close orientations, the VAE may output as reconstruction \mathbf{q}_1 and $-\mathbf{q}_2$: the network loses some of its capacity trying to represent very sharp variations that have no physical meaning. Thus, it was decided to keep the classic MSE loss for every component.

After the training of the HGGs, 2000 grasps for each stable pose of each object are generated by sampling in each HGG latent space. These generated grasps are tested

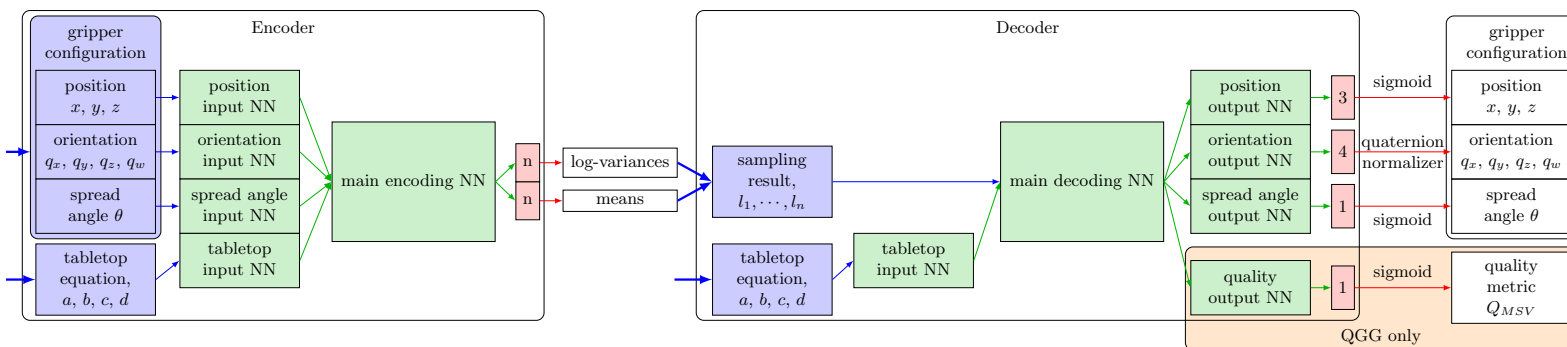


Figure 4.7 – HGG and QGG architecture. In blue the input layers, in green the hidden Neural Network (NN) and in red the output layers. The hidden NN inner layers are fully connected layers, with hyperbolic tangent activation functions. The main encoding and main decoding NN have symmetrical inner architecture. The inputs and outputs NN are small networks (with much less parameters compared to main encoding and decoding NN) in charge of extracting or reconstructing specific features associated with position, orientation, spread angle, tabletop equation or quality respectively. The supplementary input for the tabletop plane Cartesian equation ensures that the generated grasp depends on it [147]. The tabletop input NN appears both in encoder and decoder: it is the same network in both places (same weights), and not two different networks. This architecture is implemented with Tensorflow [156] and Keras [155] python libraries. The QGG has the same architecture and hyperparameters, with an added output to the decoder squared in orange.

in simulation along with primitive ones to check their success and compute their Q_{MSV} value (Equation 2.83), in order to create the extended dataset for the training of the QGGs.

A distinct QGG is trained for each object on the extended set formed by merging the primitive grasp set with the generated grasp set (both successful and failed). The architecture, hyperparameters and inputs-outputs are the same as the ones used for the HGG (see on Figure 4.7) with the grasp quality added as a supplementary output to the decoder. This way, the QGG decoder learns to predict the grasp quality while reconstructing the other grasp configuration parameters. The loss function and optimizer are also the same as the ones used for the HGG, with a MSE corresponding to the grasp quality added to the loss.

4.2 Hyperparameters Tuning and Latent Space Analysis

In the following, an interpretation and visualisation of the latent space of the HGG is presented. Then, the choice of hyperparameters for optimal grasp space modeling is discussed.

4.2.1 HGG Latent Space: Illustration in the Case of Two Latent Variables

The HGG learns to model the grasp space in its latent space. By sampling values in it, one can generate new gripper configurations that are likely to belong to the grasp space. To better understand and visualize this, some learning trials have been conducted with a two dimensional latent space having two latent variables l_1 and l_2 .

The obtained gripper configurations when exploring a two dimensional latent space for one stable pose of the bent pipe are shown in Figure 4.8. As some configurations may not lead to successful grasps, or may be in collision with the object or the environment, only pre-grasp configuration are shown (that is before closing the fingers), with collisions disabled.

A colormap of the latent space from which the gripper configurations of Figure 4.8 are extracted is displayed in Figure 4.9.

On the top-right corner of Figure 4.8 appears a configuration corresponding to the bent pipe primitive grasp type 1 (shown in Figure 4.6). The rest of the right side configurations corresponds to the grasp type 3, and the left side configurations to the grasp type 2.

It can be deduced by observing Figure 4.8 that the latent variable displayed on the horizontal axis (that corresponds to l_1 in Figure 4.9) encodes mainly the direction from which the bent pipe will be grasped: the two rearrangeable fingers on the concave side or on the convex side. The other latent variable (l_2 in Figure 4.9) is encoding mainly translations. This is more visible in Figure 4.9. However, this figure shows that the variables of the configuration space are strongly entangled in the two latent variables: for example, the three translations are controlled by both l_1 and l_2 . It may reflect an

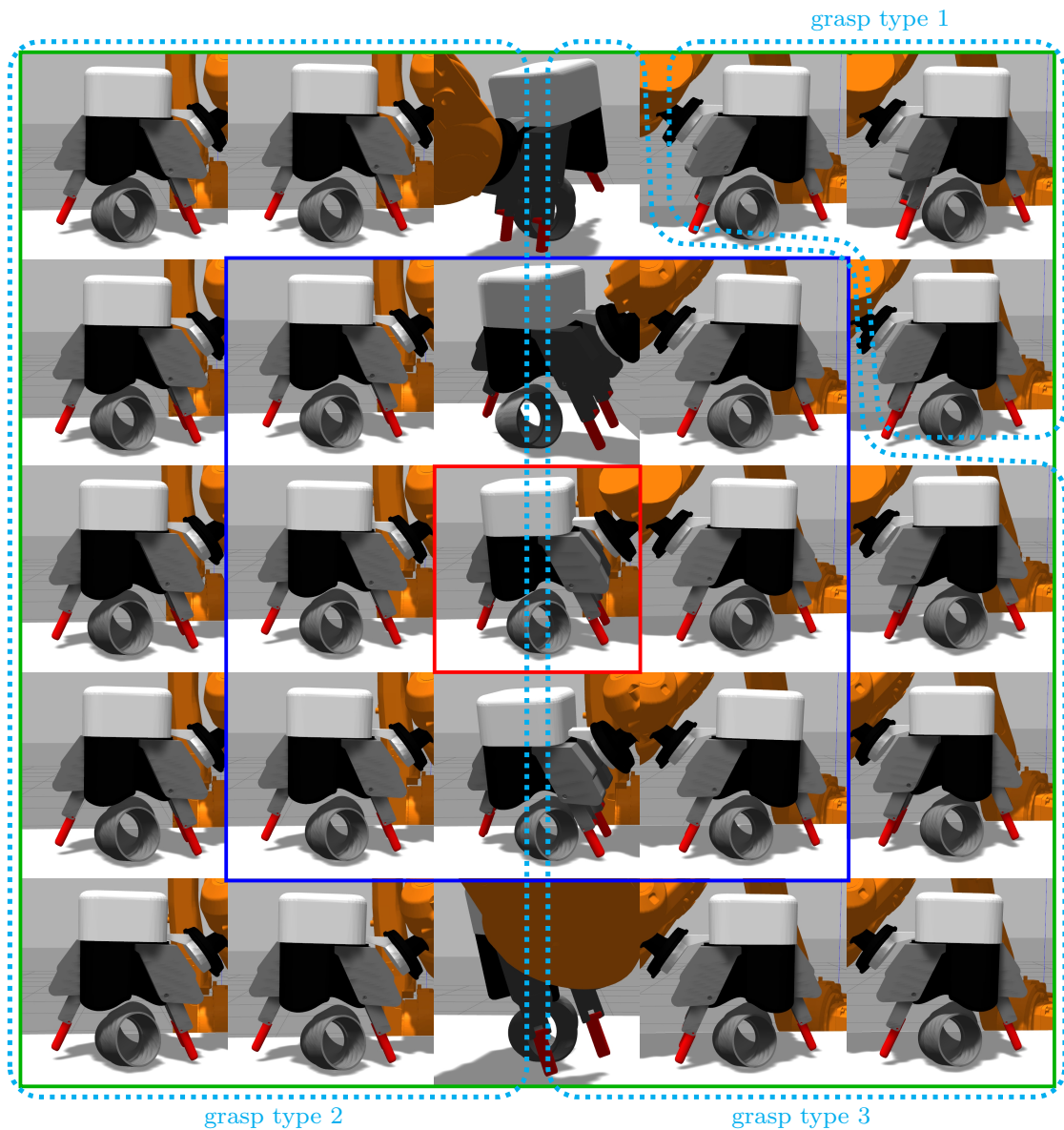


Figure 4.8 – Gripper configurations generated when visiting a two dimensional latent space of an HGG for the bent pipe. The image inside the red square is the point $(0, 0)$ in latent space. The images between the red and blue square and the images between the blue and green square correspond to points evenly distributed on circles of diameter respectively 0.5 and 1 in latent space. The coordinates of these configurations are shown with black cross markers on Figure 4.9. Here, translations along the image plan normal are not visible, which explains some visually almost identical configurations.

intrinsic property of the manifold structure of the grasp space: the translations may be highly correlated to each other and to the orientation. It may also indicate that this

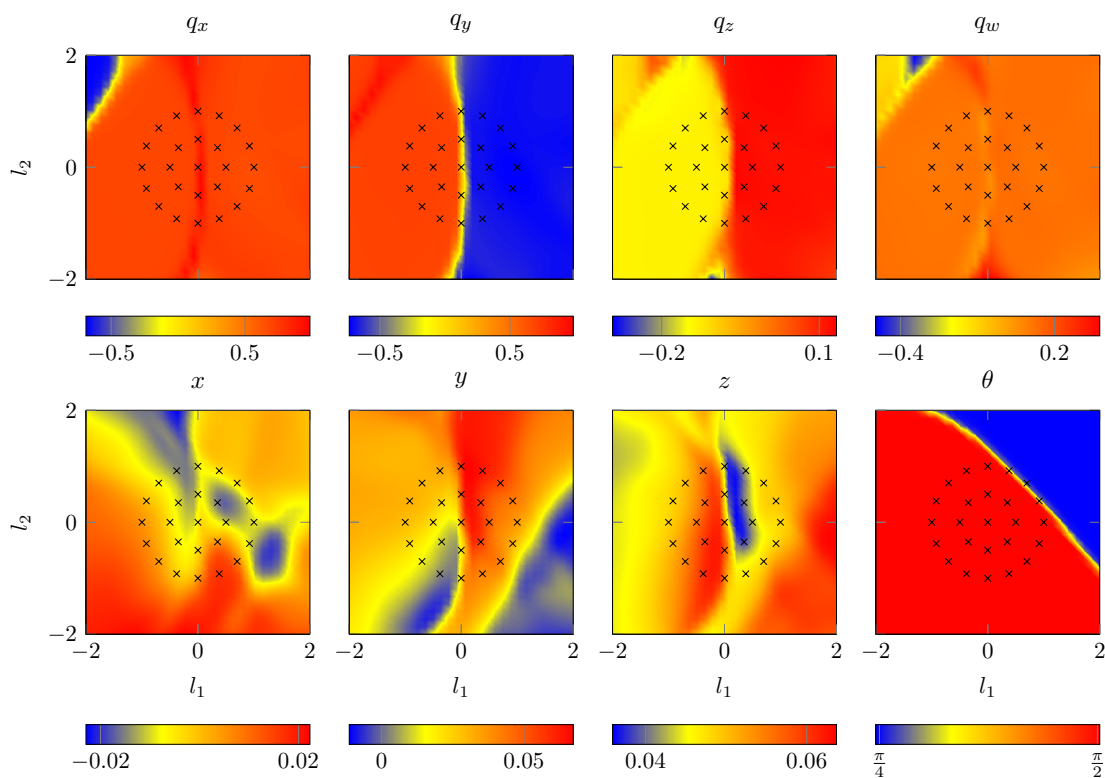


Figure 4.9 – Colormap of the latent space of an HGG for the bent pipe object, for one stable pose, when using a two dimensional latent space (latent variables l_1 and l_2). The black cross markers correspond to coordinates of configurations displayed on Figure 4.8. The color value is the output of the HGG decoder for the corresponding l_1 and l_2 values. Values for x , y and z components are in meters (and without dimension for the other components).

latent space is too small to both reconstruct accurately the training data and find a disentangled latent representation: the grasp space may be projected into a too small latent space.

Thus, the latent space dimensions, and other hyperparameters, need to be tuned carefully to obtain the best grasp space model possible.

4.2.2 HGG Tuning to Model Efficiently the Grasp Space

The HGG has three main hyperparameters that can be tuned to improve the learnt grasp space model:

- the network size;
- the latent space dimension, that is the number of latent variables;

- the KL divergence loss component coefficient [145].

Several indicators can be monitored to assess the influence of those hyperparameters on the performances of the HGG:

- the reconstruction error;
- the KL divergence loss component value reached at the end of the training;
- the number of used latent variables, that is the number of latent variables with a high KL divergence among the available latent variables;
- the proportion of generated successful grasps.

Various learning trials have been conducted with different hyperparameters combinations. A summary of the effects of the hyperparameters is given in the following subsections.

Trade-Off Between Reconstruction and Regularity

One of the distinctive features of a VAE is that its loss function combines a reconstruction loss and a regularization loss (the KL divergence), as shown in Equation 3.18. This leads the training process to converge to a trade-off between reconstruction and regularity as shown in subsection 3.3.3. Recall that:

- the reconstruction is the ability to accurately reproduce on the output the input data;
- the regularity is the fact that the input data are homogeneously distributed in each latent variable (here, following a normal distribution), and that latent variables are disentangled.

A side effect of the KL divergence constraints is that the network is pushed to use as few latent variables as possible to represent the data. For the HGG, both terms are important: a good reconstruction is needed as it allows to capture faithfully all the primitive data variability, and a good regularity is also needed as it reduces the data distribution sparsity, and thus the risk of generating inconsistent gripper configurations. **Thus, in our case, the KL divergence coefficient regulates a trade-off between the accuracy of the primitive grasp reconstruction and the proportion of successful generated grasps.**

Increasing this coefficient will put higher priority on the KL divergence term, and thus increases the regularity at the expense of the reconstruction. A too high coefficient on this term can push the network to ignore the data variability along a given axis to homogenize the data in its latent variables and disentangle them, leading to poor reconstruction. Conversely, a too low coefficient will allow a very accurate reconstruction, but the latent variables will be more entangled and their data distribution will be sparser.

The optimal value of the coefficient depends mainly on the latent space dimension. In Higgins et al. [145], they recommend a value greater than 1, but they use the VAE for image generation, which involves a latent space of greater dimension than for the presented use case. For the HGG, it has been observed that a value below 1 is mandatory to keep an acceptable reconstruction loss, typically between 10^{-2} and 10^{-4} .

Network Size

To increase the reconstruction with less impact on the regularity than the KL divergence coefficient, one can increase the network size, that is the number of neurons in the different layers, or the number of layers. Indeed, it increases the number of network trainable parameters, and thus the complexity of the functions that it is able to approximate. However, it increases the computational cost of both the learning phase and the inference phase, and the memory required to store the model. Moreover, the more parameters the network has, the more training data it needs for a proper training. For the HGG use case, there are very few training data, which limits the size of the network.

For the architecture presented in Figure 4.7, preliminary tests have shown that the reconstruction starts to improve less significantly beyond a threshold of around 30 000 parameters for the whole VAE, that is both encoder and decoder. More extensive tests would be required to determine more precisely an optimal trade-off between computational cost and general performances. In this work, the focus is put more on studying the influence of the KL divergence coefficient and latent space dimension and finding their optimal value.

Grasp Space Dimension & Latent Space Dimension

The grasp space is a subset of the gripper configuration space (see subsection 1.4.1). Thus, it has at most 7 dimensions, but its true size is a priori unknown. As the goal is to map the grasp space in the HGG latent space, it is important that the number of latent variables used by the HGG among the available ones is at least equal to the grasp space dimension. Otherwise, there will be information loss due to the compression caused by the projection of the grasp space into a smaller space. Although conservative, it is sub-optimal to let the HGG finds by itself the required number of latent variables needed to map the grasp space, by letting the latent space dimension be equal to the one of the gripper configuration space. Indeed, even if the KL divergence term in the cost function will push the network to use as few latent variables as possible, increasing the number of available latent variables increases the chances to converge toward a cost function local minimum where the network uses more latent variables than needed. Using more latent variables among the available ones also leads to a more sparse data distribution in latent space, which has the same effect as a too low coefficient on the KL divergence. **Thus, the latent space dimension also have an influence on the trade-off between the fidelity of the primitive grasp reconstruction and the proportion of successful generated grasps.**

Thus, it is useful to know an approximation of the dimension of the grasp space.

Here, a dimensional analysis tool has been chosen, the kernel PCA [107], one of the non-linear dimensionality reduction techniques described in section 3.2. The kernel PCA implemented in scikit-learn is used [157]. The algorithm is run for each object, taking as input the list of gripper configurations in the primitive grasp dataset, scaled and centered beforehand. Determining the smallest number of dimensions required to represent the data is not as straightforward with kernel PCA as with classic PCA. Indeed, the method based on eigenvalues cannot be used. To assess the number of dimension to keep, the kernel PCA reconstruction error is assessed with a number of kept dimensions from one to seven (the dimensionality of the configuration space). The smallest number of dimensions allowing to achieve a reconstruction error at most twice the one obtained with seven dimensions is considered as the smallest number of dimensions required to represent the data. The goal is to set a limit to the relative amount of information lost by the compression.

Applied on a classic PCA, this threshold select a number of dimensions corresponding to 94%, 96%, and 98% of explained variances for the bent pipe, pulley and cinder block respectively. Thus, the chosen criterion allows to keep the majority of the information. However, it is worth noting that with kernel PCA, the pre-image of a point (that is the inverse transform, which allows to compute the reconstruction error) is an approximation, computed through a ridge regression [111] in the scikit-learn implementation. Thus, this approximation contributes to the reconstruction error, and it is not trivial to discriminate between the contribution of the pre-image approximation and the contribution of the dimension reduction.

	PCA	kernel PCA
bent pipe	6	5
pulley	5	3
cinder block	7	5

Table 4.1 – Smallest number of dimensions allowing to achieve a reconstruction error at most twice the one obtained with seven dimensions.

The result of this analysis for the three tested objects is summarized in Table 4.1. The fact that kernel PCA is able to find a smaller number of dimensions than classic PCA confirms the non linear nature of the relationships existing between the configuration space and the grasp space. It can be observed that the number of dimensions depends on the object: the complexity of the grasp space depends on the chosen primitive grasps, and then on the object geometry. It is intriguing that the number of dimensions found with kernel PCA corresponds for each object to their number of primitive grasp types (respectively 5 primitive grasp types for the bent pipe and cinder block, and 3 for the pulley). The most straightforward way for kernel PCA to organize that data may be to dispatch the primitive grasps corresponding to each grasp type along a different principal component in the feature space. If this interpretation is correct, this representation may be very sparse and may not be the most compact, as the majority of the produced space

may not correspond to any data point, such as areas not aligned on a given principal component.

Therefore, this can serve as an upper bound for the optimal latent space dimension. Indeed, the HGG, being a generative model, will be forced to find a dense representation, which may also be more compact than the one found by kernel PCA. Moreover, it has a supplementary information: the tabletop Cartesian equation, and may use it to better organize the data in the latent space, and learn a more compact representation.

Overview

Table 4.2 summarizes the influence of the three hyperparameters on the chosen indicators, assessed through the computation of correlation coefficients between the hyperparameters and the indicators. For this evaluation, trials were conducted with hyperparameter combinations among the following ranges:

- network size between 12 000 and 30 000 parameters;
- latent space dimension between 2 and 6;
- KL divergence coefficient between 0.0002 and 0.01.

	latent space dimension	KL divergence coefficient	network size
number of used latent variables	0.12	-0.26	0.55
reconstruction error	-0.18	0.18	-0.21
KL divergence	-0.75	-0.62	0.09
proportion of generated successful grasps	-0.39	0.25	0.39

Table 4.2 – Spearman correlation coefficients between the hyperparameters and the indicators.

The reconstruction errors corresponding to the set of hyperparameters achieving the best trade-off between the correct reconstruction of primitive grasps and the proportion of successful generated grasps are shown in Table 4.3.

Some statistics about primitive grasps and grasps generated with this hyperparameter set are summarized on Table 4.4. To avoid arm kinematic reachability issues, as gripper configurations are in object frame, each generated configuration is tested for different object orientations relative to the robot. The main cases of failing grasps are found when transitioning between different grasp types, and with the fifth grasp type of the bent pipe

	mean position error (m)	mean orientation error (degree)
bent pipe	0.004	1.94
pulley	0.005	1.32
cinder block	0.009	1.1

Table 4.3 – performances of the HGG for the selected hyperparameters: 30 000 network parameters, 3 latent variables (that are all used), and a KL divergence coefficient of 0.0005. The mean errors are measured on the training data.

		generated set	primitive set	
bent pipe	total number of grasps	4000	145	
	number of successful grasps	2727 (68.18%)	141	
	metric statistics	median	0.0954	0.1018
		mean	0.0982	0.1047
		maximum	0.2066	0.2257
	cinder block	total number of grasps	6000	141
		number of successful grasps	5608 (93.47%)	141
metric statistics		median	0.0684	0.0670
		mean	0.0580	0.0564
		maximum	0.1401	0.1041
pulley		total number of grasps	4000	118
		number of successful grasps	3367 (84.18%)	111
	metric statistics	median	0.0702	0.0730
		mean	0.0652	0.0648
		maximum	0.1420	0.1195

Table 4.4 – Information summary about primitive and HGG generated grasps (step 3 of the workflow in Figure 3.14) for the selected hyperparameters: 30 000 network parameters, 3 latent variables (that are all used), and a KL divergence coefficient of 0.0005. The metric statistics are taking into account successful grasps only.

(Figure 4.6, top right) where one of the bottom fingers can collide with the table in the pre-grasp phase for some gripper orientation variations.

The grasp quality mean and median of the primitive and generated grasps are close to each other. This is expected as the VAE tries to reproduce the underlying distribution of the learning set.

For two objects, a quality metric global maximum better than the primitive grasps is found in the generated grasps. Indeed, the VAE learns to interpolate between the primitive grasps: in case the parameters defining a grasp configuration with higher metric value are close to parameters of two primitive grasps, it is able to generate it.

Regarding the bent pipe generated grasps, none of them are better than the best primitive grasp. As no significant differences were noted at the training step in term of reconstruction and regularity performances between this object and the others, this is unlikely to be due to a poor grasp space modeling by the bent pipe HGG. Two possible explanations remain:

- The global maximum may already be in the primitive dataset, but the random sampling in latent space may fail to draw a sample sufficiently close to this maximum when generating new grasps. It is not unlikely, as it is a human-crafted set of configurations, and humans tend to produce high quality grasps.
- The chosen quality metric may have very little local variations on this particular object, given its geometry. A small change in the grasp configuration, which occurs when exploring the grasp space around primitive configurations, may lead to no significant changes in the metric value. Thus, the maximum found among primitives may be some kind of outlier (due to the noisiness of the contact simulation) in a plateau-like area.

4.3 Experiments

To validate the grasp space exploration workflow, the proposed method is compared to a sampling-based exploration method, and grasp planning trials are conducted firstly in a simulated environment, and then on a real setup.

4.3.1 Comparison with Classic Sampling-Based Grasp Space Exploration Methods

As stated in section 3.4, the proposed method focuses on the grasp space exploration, which is a necessary step for any grasp planner algorithm. For learning based grasp planner for example, the grasp space exploration step corresponds to the constitution of a grasp dataset that will serve as training data for the learning algorithm. Thus, in this section, our method is compared to some methods classically used in other works dealing with grasp planning to create their learning dataset.

For underactuated grippers, the grasp space exploration can be done only with a gripper configuration approach, and not with a contact point approach, due to the nature of the underactuation mechanism as explained in subsection 1.4.2. Gripper configuration approaches are mainly divided in two categories, depending on the considered configuration space.

grasp space exploration method	hypothesis on grasp space	target gripper architecture	reported grasp sampling success rate
planar methods, used for example in: Pinto and Gupta [68], Depierre et al. [69], Levine et al. [70]	Gripper configurations aligned with a plane, described by three parameters: position x, y and angle θ in the plane	parallel jaws grippers	$\approx 10\%$ (8.05% in [68], 10% in [70])
6-DoF methods, used for example in: Riedlinger et al. [57], Mousavian et al. [71]	Gripper configurations expressed relative to mesh vertex normals, by two [71] or four [57] parameters	parallel jaws grippers	$\approx 10\%$ (1.49% in [57], 19.4% in [71])
our method, using QGG network	gripper configuration expressed in the Euclidean space, compressed on three parameters, taking inspiration from human grasp strategies	multifingered and underactuated grippers	Depends on the object and on the chosen quality threshold, in the worst case (no threshold), the success rate is the one of the HGG, $> 68\%$

Table 4.5 – Comparison of main characteristics of grasp space exploration methods.

- planar methods [68–70]: only vertical pinch grasps are considered, and the gripper is constrained to stay aligned with a vertical axis. This is a relatively strong hypothesis, and is suited only for parallel jaws grippers. A gripper configuration is parameterized in the plane by three variables : the position x, y , and the orientation θ . Some variations of this approach exist, such as in Levine et al. [70], where the authors also take into account the gripper trajectory before the grasp in a visual servoing framework, and model a grasp as a set of gripper motions. For these methods, grasps are generated by sampling randomly in the parameter space. This sampling can be constrained using features detected in an image, for example by sampling configurations only in some region of interests or only near parallel edges.
- 6-Dof methods [57, 71]: a grasp configuration corresponds to a pose in $SE(3)$. In

practice, this type of method is mainly used for parallel jaw grippers, although in theory it could be extended to grippers with more degrees of freedom. The main assumption made by this approach is that valid grasps are more likely to be found when the grasp direction axis of the gripper is facing the object and aligned with its surface normal. To sample grasps based on this assumption, a discretized object representation is used, for example a mesh or a point cloud. The main idea is to sample vertices on the object surface, and generate gripper configurations relative to the sampled vertices and their normal vectors. These gripper configurations can be generated for example by sampling several rotation angles around the normal axis, and several distance from the object surface [71]. Additional degrees of freedom can also be added, such as sampling around the vertex normal axis in a conical fashion [57].

Characteristics of both approaches are summarized in Table 4.5, and compared with our grasp space exploration method.

However, it is difficult to compare the reported success rate of different methods. Indeed, other methods are very often applied on parallel jaws grippers, which have a lower dimensional internal configuration space than multifingered grippers. To better compare sampling-based grasp space exploration methods with ours, one of them is implemented and tested on our simulated setup.

Our approach being able to generate grasp in the Euclidean space, it is closer to 6-DoF methods than planar methods. The chosen method for comparison is similar to the 6-DoF method used in Mousavian et al. [71] to create their training dataset. This method is chosen because it is simple to implement, and a relatively high success rate is reported in the original work. With the chosen method, grasps are sampled as follows:

1. sample a vertex of the object mesh
2. align the gripper palm normal with the vertex normal
3. sample a distance between the gripper palm and the object surface
4. sample a rotation around the gripper palm normal
5. sample an abduction-adduction angle

For each stable position of each object, 5000 grasps are sampled and tested in the simulation, in a similar way as the HGG generated grasp in the step 3 of our workflow. In Table 4.6 are summarized the results of these simulated trials.

The success rate is lower than the one reported in Mousavian et al. [71] for two reasons.

- Firstly, in Mousavian et al. [71] grasps are sampled around a floating object. In our implementation, collisions with the surface on which the object is standing are considered.

	success rate (%)	median quality metric	mean quality metric
bent pipe	4.15	0.0668	0.0612
cinder block	5.92	0.0565	0.0500
pulley	4.78	0.0528	0.0488

Table 4.6 – performances summary of a sampling-based grasp space exploration method. The mean and median quality metric values are computed on the successful grasps only.

- Moreover, in our implementation, a supplementary sampling dimension is added because of the abduction-adduction degree of freedom, which can reduce even further the success rate. Indeed, for a given gripper spatial configuration, only some abduction-adduction angles are suitable for a proper grasping. This indicates that this type of grasp space exploration methods is not well suited to grippers with a high dimensional configuration space.

The grasp success rate of this classic sampling-based method is one order of magnitude smaller than the grasp success rate of grasps generated by the HGG (see Table 4.4), which roughly corresponds to the worst case scenario of sampling grasps with the QGG without filtering grasps based on a threshold on the predicted grasp quality. Moreover, the mean and median quality metric values of the few successful grasps are also smaller than the one measured on grasps generated with the HGG (see Table 4.4).

This shows that our approach allows to reduce drastically the number of simulation trials required to explore the grasp space. It also shows that focusing the search around human provided grasp strategies produces qualitatively better grasps than searching randomly in the configuration space.

4.3.2 Simulation Trials

Grasp planning trials are conducted in Gazebo on each object. The algorithm used to choose a grasp is described in Algorithm 1.

This planning procedure is executed on 1000 distinct object poses for each stable position of each object. The position of the object frame projection on the tabletop plane is chosen randomly inside a 10×10 centimeters square, and its orientation relative to the vertical axis is also drawn randomly between 0 and 2π .

Three metrics are monitored to assess the performances of the presented workflow:

1. the grasp success rate.
2. the number of collision and reachability checking iterations needed to find three admissible grasps (Algorithm 1 line 5), as it is the most time consuming step. Indeed, the presented workflow is object-centric. It does not take into account the arm kinematics and environment, so depending on the object pose in the robot

Algorithm 1 Grasp planning algorithm.

```

1: grasp candidate list  $\leftarrow \emptyset$ 
2: while length(grasp candidate list) < 3 do
3:   configuration  $\leftarrow$  QGG decoding of a sampled value in its latent space
4:   if configuration predicted grasp quality > threshold then
5:     if configuration is collision free and kinematically reachable then
6:       append configuration to grasp candidate list
7:     end if
8:   end if
9: end while
10: execute the grasp with highest predicted grasp quality among grasp candidate list

```

workspace, the probability of sampling an admissible configuration in the QGG latent space varies.

- the grasp quality prediction relative error.

These performance metrics are shown in Table 4.7.

The grasp quality prediction mean relative error is quite high. Indeed, due to contact points location variability in simulation, the computed grasp quality metric has a variability and is not fully deterministic for a given configuration, which makes it noisy. Thus, this parameter is difficult to learn and prone to underfitting or overfitting issues, especially as each QGG is trained on only 2000 samples for each object stable pose.

The contact points location variability is caused by a slight and constant oscillation of the simulated finger underactuated joints. This slight finger oscillation also makes the simulated grasps less stable than in the real world. This oscillation is most likely caused by a numerical instability in the simulation of the gripper underactuation, more specifically of its integrated passive spring. Indeed, the amplitude of the oscillations are correlated with the spring stiffness, and disappear when deactivating the simulation of the underactuated system. It is suspected that this instability comes directly from the internal computations of the Gazebo physics engine. Indeed, it uses internally a semi-explicit integration method, which is known to generate numerical instability with elastic forces. Finding the exact root cause and fixing it probably requires an in depth

	1) success rate (%)	2) Algorithm 1 line 5 mean iterations	3) mean quality prediction error (%)
bent pipe	99.75	5.7	18.9
cinder block	99.37	4.9	13.8
pulley	99.6	6.8	17.1

Table 4.7 – Performances on simulated grasp planning trials.

investigation of the physics engine, which is out of the scope of this work.

The low number of collision and reachability checking iterations shows that all grasp types and their variants are evenly distributed in the latent space. Indeed, some grasp types or variants within a grasp type are reachable only for some object poses relative to the robot. This also shows that the majority of configurations colliding with the table are correctly labeled with a low quality value. Indeed, if these configurations were not mostly eliminated by the grasp quality threshold at the line 4 of Algorithm 1, they would be eliminated at line 5, thus would lead to a high number of collision and reachability checking.

The low failure rate shows that the procedure presented in this work successfully explores and reproduces the grasp space, as it is able to generate reliably successful grasps for various object poses, despite a relatively high prediction error on the grasp quality. The few grasp failures are due to critical grasp quality prediction errors, more likely to occur on areas of the grasp space that are at the border between successful and failing grasps. In these areas, very sharp variations of the grasp quality prediction are required, whereas the trade-off between reconstruction and regularity induced by the KL divergence prevents such variations.

4.3.3 Physical Trials

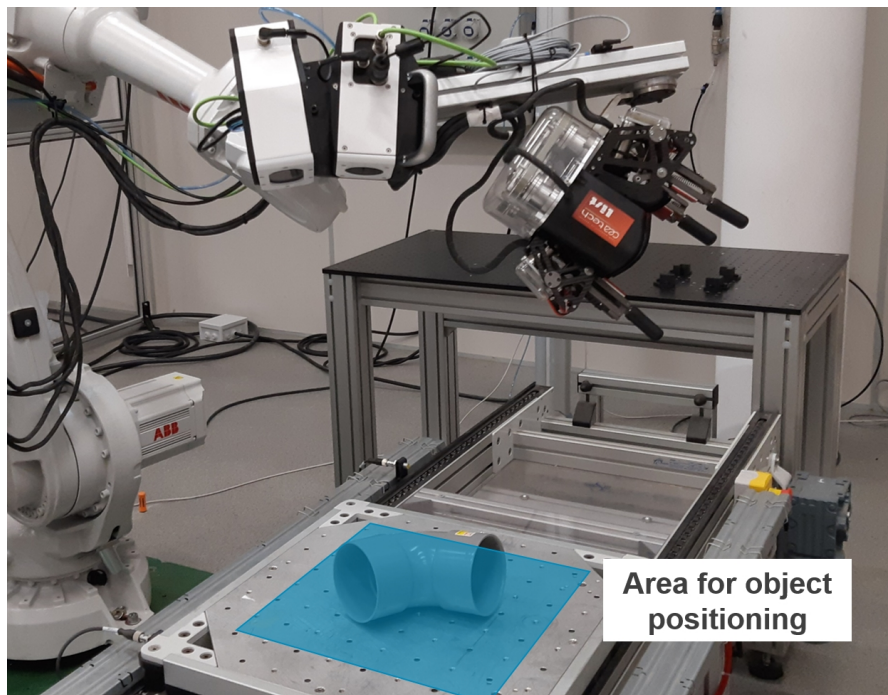


Figure 4.10 – Area for object positioning. For each trial, the object pose is chosen randomly inside the blue rectangle.

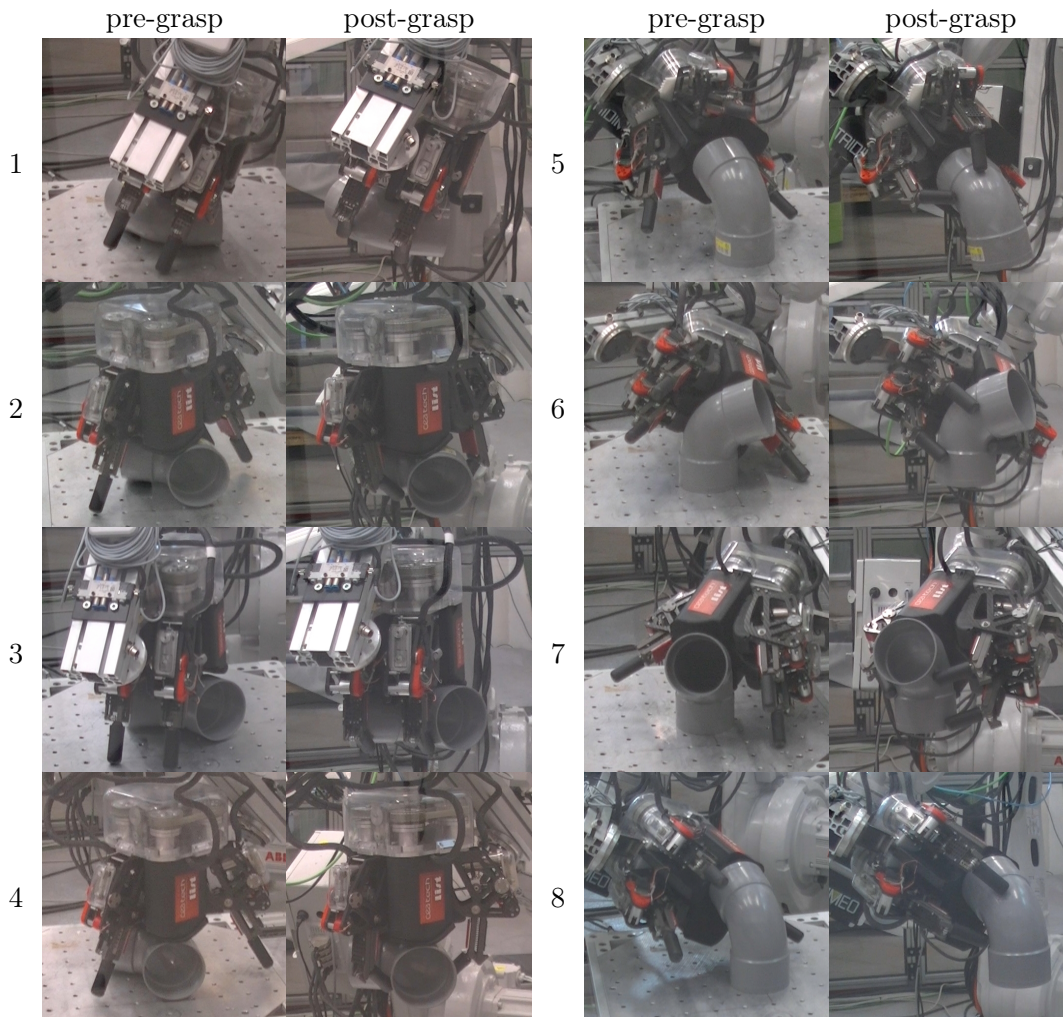


Figure 4.11 – Pictures of eight grasps generated by the QGG and executed on the real setup after following the grasp planning procedure described in Algorithm 1.

Grasp planning trials are also conducted on the real setup described in subsection 4.1.1 on the bent pipe. The grasp planning algorithm used to choose a grasp is still the one described in Algorithm 1.

About thirty grasp trials have been conducted with an equal distribution between the two stable poses of the bent pipe. For each trial, the object is placed randomly in an area of the workspace shown on Figure 4.10 by an operator.

Every generated grasp succeeded to lift the piece from the table, and to maintain it in the hand. The only failure cases were noted when the camera and associated algorithm failed to retrieve correctly the object pose. In that case, as the object pose is unknown, the trial cannot be conducted. This occurs for example when the object is too far from the camera, or when a significant part of the object is off-camera. This issue can be fixed

by trying a different camera point of view when the object pose retrieval fails.

Eight examples of grasps conducted on the real setup are displayed in Figure 4.11. In this figure, the grasps 1 & 2 correspond to primitive grasp type 3 on the top row of Figure 4.6, the grasp 3 to primitive grasp type 2, the grasp 4 to the primitive grasp type 1, the grasp 5 to primitive grasp type 5 and the grasps 6 & 7 to primitive grasp type 4. The grasp 1 is tilted relative to the tabletop plane. Such a grasp differs significantly from the primitive grasp variations manually specified for the primitive grasp type 3. Generating this kind of variations can help the kinematic reachability along for some unfavorable object poses, or in cluttered environment. The grasp 8 can be seen as a fusion or an intermediate state between the primitive grasp type 4 and type 5, and does not correspond to any primitive grasp. This shows that the presented method is able to find novel and efficient way to grasp an object.

4.4 Conclusion

In this chapter, the performances of the grasp space exploration method proposed in this work have been assessed. For that, it has been applied on three objects, using a three-fingered underactuated gripper designed at CEA robotic laboratory.

First, this method has been compared with other grasp space exploration methods based on random sampling. One of these sampling based exploration methods has been implemented and tested in simulation to better compare it with ours. This comparison shows that our method allows to explore the grasp space much more efficiently: our method produces more than ten times more successful grasps, and the produced grasps have higher grasp quality metric values on average.

Then, grasp planning tests have been conducted in simulation, as well as in real conditions. These tests show that the proposed method is able to generate reliably successful grasps with a correct estimation of their quality, can be transferred successfully on a real robotic setup, and is able to generate novel grasps that differ from the one specified as primitives.

Chapter 5

Extension Towards Object Geometry Information

This chapter presents a variant of the grasp space exploration method introduced in chapter 3, and experimentally qualified in chapter 4. The main drawback of the proposed method is the fact that it requires the training of a separate neural network for each individual object, and also the use of a set of primitive grasps for each of them. The variant proposed in the following aims at taking into account the object geometry in the training. This allows to simplify the workflow, by using a single neural network trained with grasp data from every known primitive objects. It allows to reach performances similar to the ones reached with the standard QGGs. Experiments also suggest that it might be used to generate grasps for deformed versions of known primitives objects, but with less reliability than for the original objects. First, the chosen approach for the integration of the object geometry is described. The various attempted approaches regarding geometry processing are presented: several unsuccessful ones have been tested before finding a satisfactory method. Finally, some results on grasps generation obtained by this grasp space exploration variant are presented.

Contents

5.1	Approach	146
5.1.1	3D Data Modeling	146
5.1.2	Integration of Object Geometry Information	149
5.1.3	Object Geometry Dimension Reduction	150
5.2	Learning to Compress Object Geometry	159
5.2.1	Network Architecture and Training	159
5.2.2	Results	160
5.3	Using Object Geometry to Generate Grasps	165
5.3.1	Geom-QGG Architecture and training	165
5.3.2	Experiments	167
5.4	Conclusion	171

5.1 Approach

In this section some context on 3D data modeling is given. Then, the rationale behind the chosen approach for integrating the object geometry in the proposed grasp space exploration workflow is explained.

5.1.1 3D Data Modeling

The physical world being in 3D, we interact permanently with 3D objects. However, to be able to reason and perform computations on such structures, they need to be modeled, in a way compatible with computers. Several modeling frameworks exist, and are able to represent 3D geometries. In the following, the main existing frameworks are presented, but this list is not intended to be exhaustive.

- Firstly, Point Clouds can be used to model 3D data. It may be the most simple and lightweight model possible: the spatial coordinates of points situated on the surface of the modeled object are stored. Eventually, a normal vector associated to each point can also be computed from the point cloud and stored.

This type of models is typically output by 3D scanners, 3D cameras or lidars. However, a point cloud does not model explicitly the geometry itself: it is not trivial for example to determine if a given point is inside or outside a geometry modeled by its point cloud.

- Another particular case of 3D data modeling is solid modeling. These types of models aim at allowing the unambiguous determination of point membership to the model geometry [158, 159], on the contrary to point cloud model. Solid modeling can be divided in three main subcategories: Constructive Solid Geometry (CSG), Boundary representation (Brep), and spatial subdivision representations.
 - Constructive Solid Geometry applies a combination of set Boolean operations (union, intersection, difference) and rigid transformations (rotation, translation, homothety) on several primitive geometries, to build a more complex

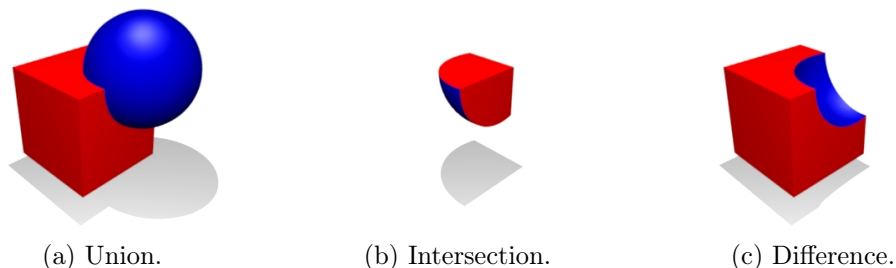


Figure 5.1 – Examples of set Boolean operations used in CSG on a block and sphere primitive shapes. pictures: CC-BY-SA license [160]

structure [161]. An example of set Boolean operations applied on two primitives is displayed in Figure 5.1. The primitive shapes typically used are block, sphere, cylinder, cone, and torus.

A geometry can be represented by a tree structure that has primitive shapes as leaves and performed operations as nodes. Most CAD softwares are based on this representation. The main advantage of this representation is that it allows an exact representation of the modeled geometry, and that testing the watertightness of a shape is straightforward. The main drawback is that for complex shapes, the CSG representation can involve a high number of operations and primitives, and the tree representing the shape can become extremely large. Moreover, depending on the considered set of primitives, even some simple shapes may require a substantially high number of primitives and operations to be represented accurately.

- The spatial subdivision representation, for its part, partitions a solid into a set of cells, each having a simple topological and geometrical structure. This category includes finite-element meshes, voxels and octrees. In finite-element mesh representation, the volume of an object is modeled as a set of adjacent simple polyhedras, that approximate the desired geometry. The voxels representation divides the space with a regular grid, in a similar way as pixels in a 2D image, each voxel having a value indicating if it is inside or outside of the modeled geometry. Finally, an octree partitions the space in height cubes, each cube being also divided recursively until a given cube size is reached. This subdivision can be represented in a tree structure, each node having height children. Each cube is labeled black if it is inside the geometry, white if it is outside, and grey if it contains the geometry boundary.

These representations allow to describe explicitly the volume of the modeled solid, and implicitly its boundaries. However, the size of these representations depends directly on the number of cells (mesh elements, voxels, or octree cubes) used, and thus on the accuracy of the representation. In particular, for octree and voxel representations, a significant amount of cells represent empty space, thus such representation is not very compact.

- With Boundary representation (Brep), a solid is represented by its external surface. Such description is composed of two parts: the topology and the geometry. The topology is described with faces, edges and vertices, connected to each other to form the final geometry [162]. To each element is associated a geometry as follows: a face is characterized as a bounded area of a surface, an edge is described by a portion of a curve, and a vertex is associated with a point in space.

On the contrary to spatial subdivision representation, Brep describes explicitly the boundaries of a solid, and implicitly its volume. This representation is also vastly used in CAD softwares. In the general case, Brep allows an exact representation of the modeled geometry, as CSG. Moreover, it is more

flexible than CSG, as it does not depend on a limited set of primitive geometries. However, it is a very heavy representation, as it needs to store all the parameters required to define the curves and surfaces expressions, together with their corresponding bounds.

A particular case of Brep is the mesh boundary representation. It is a different concept from finite-element mesh. It assumes planar faces, and that all faces are polygons having the same number of vertices. The mesh boundary representation can be modeled as an undirected graph, each vertex having an associated value corresponding to its position in space, as shown in Figure 5.2. A widely used representation is the triangulated mesh, with all faces being triangles.

The mesh boundary representation, also called simply mesh, is more compact than classic Brep, because the geometry of faces and edges is directly inferred from the position of their vertices: there is no need to store additional parameters describing the faces or edges geometries. Every faces having the same number of vertices, the topological description is also simplified compared to classic Brep. However, this representation is an approximation of the true geometry, the quality of the approximation depends on the number of vertices chosen to model the geometry.

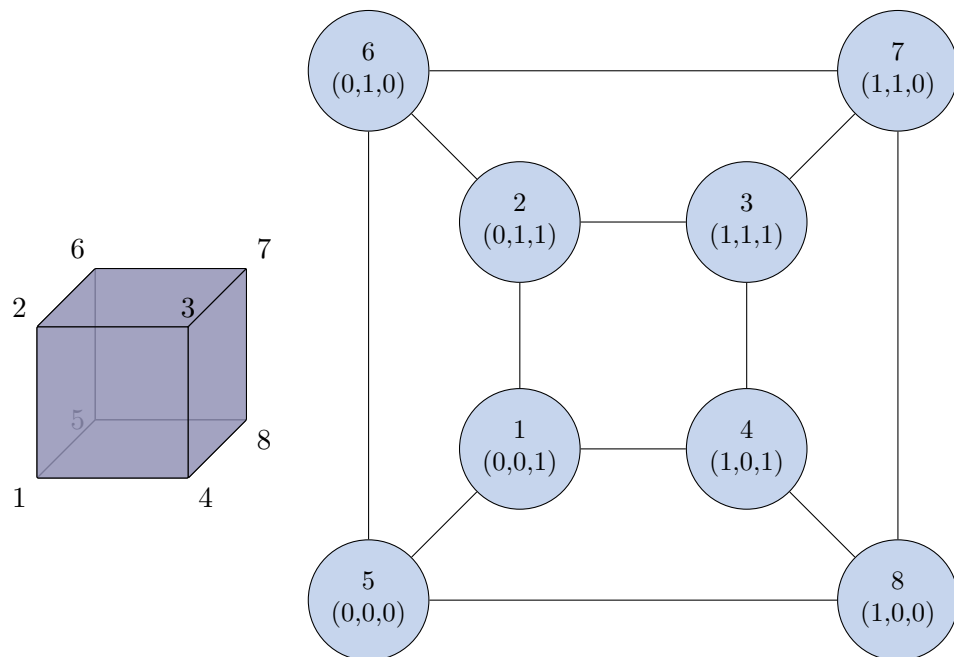


Figure 5.2 – Example of the graph of a mesh boundary representation of a cube, using square mesh faces. On the left, the mesh of the cube is depicted, with each vertex numbered. On the right, each vertex is represented as a blue node, with its corresponding spatial position between parentheses.

5.1.2 Integration of Object Geometry Information

For a given gripper, the grasp space depends on the object geometry. In the grasp space exploration method proposed in section 3.4, this dependence is implicit, as the procedure is applied separately object by object. It can be impractical for two reasons: first, it can be inefficient and complex to store and manage several neural networks when one needs to generate grasps for several objects, and then, this method cannot capitalize on known primitives to generate grasps for other known objects with unknown primitives, even if the other object resembles an object with known primitives.

Thus, taking into account object geometry information in the grasp space exploration should allow to simplify the workflow with a single QGG, that will be called geom-QGG in the following, and might allow to generalize to a certain extent to objects similar to the ones with known primitives. To achieve that, recall that the QGG is a CVAE, conditioned on the tabletop plane Cartesian equation. The grasp dependence on object geometry can be expressed in this framework as a supplementary condition in the geom-QGG. A scheme of this architecture is shown in Figure 5.3.

However, as shown in the previous subsection, the description of the geometry of a solid can be very large in term of number of parameters. Thus, if the geom-QGG is processing it as is, the risk will be that it will lose a significant part of its capacity

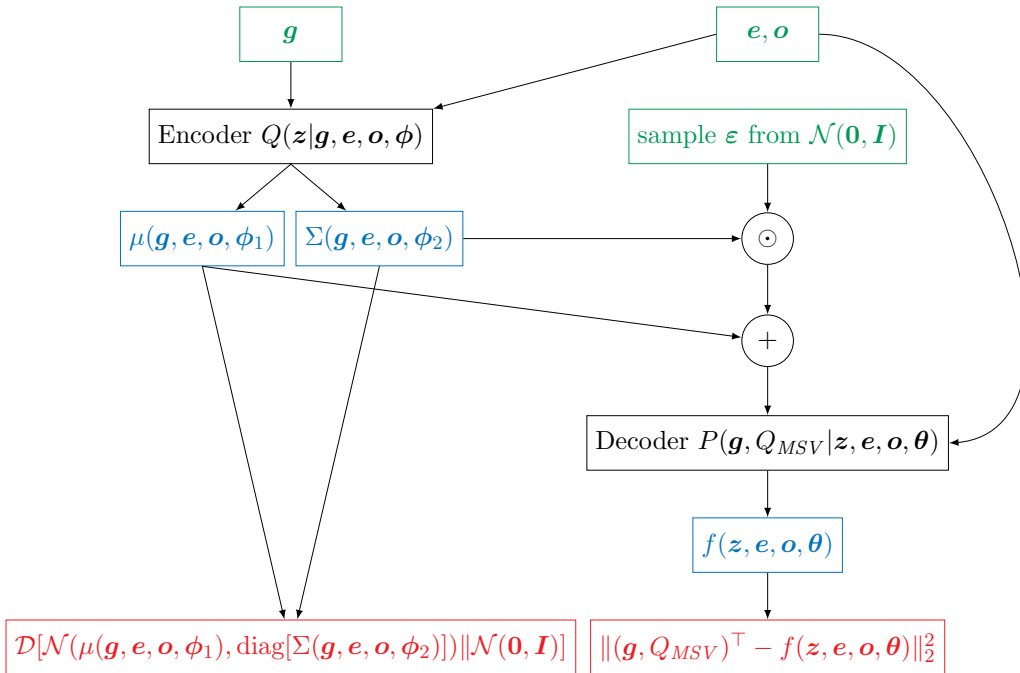


Figure 5.3 – Scheme of the training workflow of the geom-QGG, with g the grasp configuration, e the tabletop Cartesian equation, and Q_{MSV} the grasp quality as defined in subsection 3.1.2, and o a representation of the object geometry. In green the inputs, in blue the outputs, and in red the loss functions used during the training.

trying to extract information from the geometry representation, or in the worst case, will fail to extract anything. This risk is amplified by the fact that the dataset of objects with primitive grasps available is small. Thus, a preprocessing of the object geometry is required. Its objective is to extract the important features of the object geometries, or, in other words, compressing these features in a smaller space, that can then be used by the geom-QGG. The extracted features need to summarize the geometry of the object, but they should also store information on the geometry orientation and position relative to its reference frame. Indeed, the grasp configuration and the tabletop plane Cartesian equation being expressed in the object frame, this is an important information for the geom-QGG.

In recent years, deep learning approaches based on neural networks have been more and more used to extract semantic information from 3D data. For this purpose, the most commonly used input data are point clouds [163–165], meshes [166–173], and voxels [174–176]. The grid structure of voxel inputs allows to use 3DCNN, in a similar way to classic CNN used for image processing [130]. Nevertheless, the 3D structure increases the computational cost of such networks compared to the 2D case. Regarding point clouds, the lack of structure (the points being unordered) make more complex the information extraction, and special network architectures have been designed, such as PointNet [164] for example. The mesh representation for its part raises an other issue: its graph structure is not straightforward to take into account in a neural network implementation. Recently, several attempts to implement graph convolution have been published [166, 177–180], but it is still an ongoing research topic: no method seems to stand out as the established one, nor is natively available in the main machine learning libraries at the moment. One can refer to Bronstein et al. [181] for an in depth overview of the subject.

However, regardless the chosen input data representation, these architectures are often complex, and their main target tasks are object part segmentation [167, 168, 170], object classification and recognition [170, 171, 174–176] or object generation [163, 165, 166, 169, 172–174]. This does not fit directly our application, and such approaches often end up by design with an internal representation of the object geometry independent of the object frame, which is not suitable for our application.

5.1.3 Object Geometry Dimension Reduction

To reduce the dimension of the object geometry representation, one first needs to choose the type of representation to take as input. Here, it is decided to work with a triangular mesh. Indeed, it is one of the most commonly available format, and one of the most compact if the number of vertices is chosen appropriately. For example, some mesh dataset are available, such as ShapeNet [182].

Then, for the dimension reduction itself, inspiration is taken from works dealing with geometry generation [163, 165, 166, 169, 172–174] that often use an autoencoder structure to produce a low dimensional latent space corresponding to object geometries. However, with these methods, the produced latent space is often very large, on the order of one hundred latent variables. For the geom-QGG, such additional feature vector would be significantly larger than the other inputs (the gripper configuration and the tabletop

Cartesian equation), and may still be an issue for a proper learning.

The proposed architecture is summarized in Figure 5.4. A mesh preprocessing step (step 1) allowing to reduce the size of the input of the geometry autoencoder (step 2) is required, so that the geometry autoencoder can produce a lower dimensional latent space (step 3), that the geom-QGG (step 4) can handle more easily. Moreover, as a mesh can have an arbitrary number of vertices, the mesh preprocessing must deal with this possible variation of the number of vertices, so that the geometry autoencoder has an input of constant size. Regarding this issue, works dealing with mesh generation assumes a constant number of vertices.

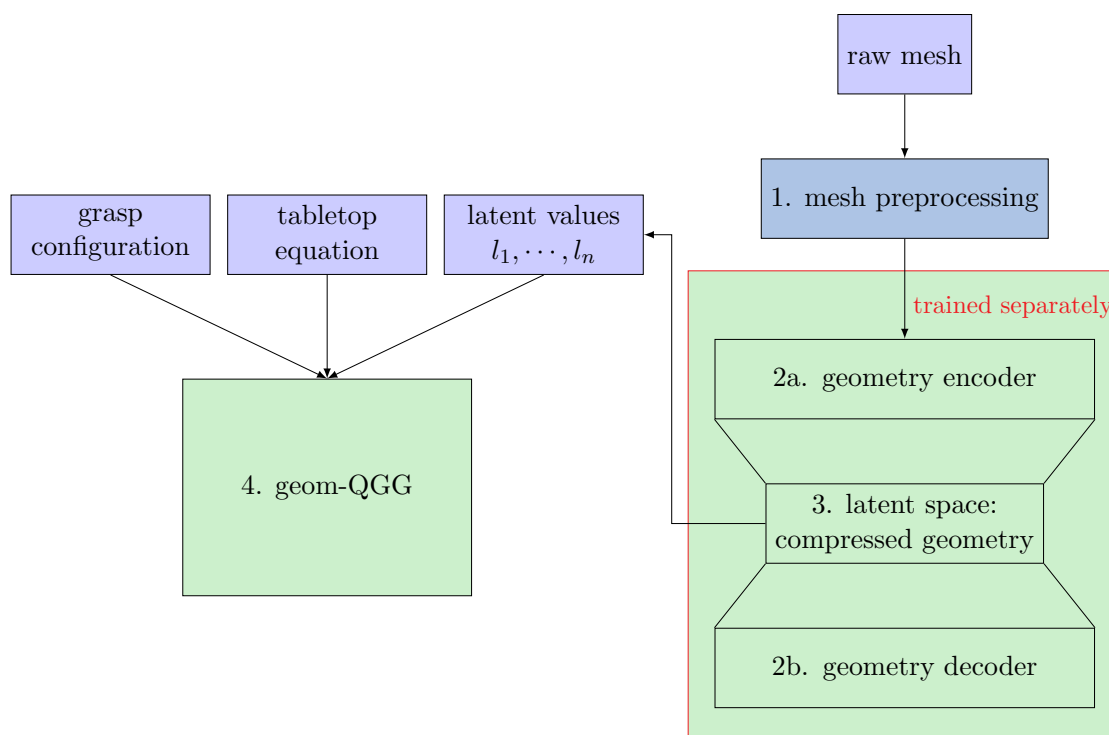


Figure 5.4 – Scheme of the proposed workflow to take into account the object geometry in the grasp space exploration.

Several approaches have been tested and found to be unsuccessful for the mesh preprocessing step. They are described in the following, together with the approach that has finally allowed to reach the desired results.

Spectral Clustering Segmentation

The first attempted idea to preprocess the mesh to reduce the size of the representation was to segment it into a small and fixed number of parts, and then summarize the segmented geometry with a few parameters. An unsupervised way to perform object part segmentation from its mesh representation is spectral clustering [183]. It relies on the

notion of graph Laplacian matrix. There is no strict definition of the graph Laplacian, and several variants exist. For an arbitrary graph having n nodes, the Laplacian $\mathbf{L} \in \mathbb{R}^{n \times n}$ can be defined as:

$$\mathbf{L} = \mathbf{D} - \mathbf{A} \quad (5.1)$$

with \mathbf{A} the adjacency matrix, a symmetric matrix having coefficients $a_{ij} = 1$ if node i and j share an edge, and $a_{ij} = 0$ otherwise, and \mathbf{D} the degree matrix, a diagonal matrix with coefficients $d_{ii} = \sum_j a_{ij}$. In Figure 5.5 is shown an example of Laplacian matrix computation for a simple graph.

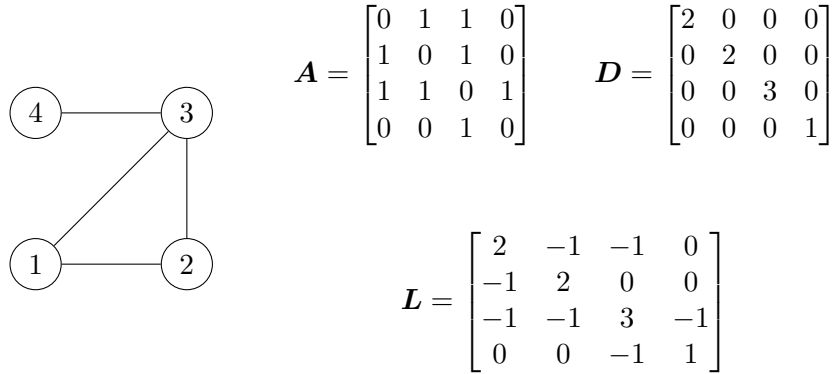


Figure 5.5 – Adjacency, degree and Laplacian matrices of a graph, noted \mathbf{A} , \mathbf{D} and \mathbf{L} respectively.

For a mesh, the definition of the Laplacian matrix can be modified so that it becomes the discrete approximation of the laplace-Beltrami operator over a surface [184]:

$$\mathbf{L} = \mathbf{V}^{-1}(\mathbf{D} - \mathbf{W}) \quad (5.2)$$

with \mathbf{V} a diagonal matrix with coefficients V_{ii} representing vertex weights defined as the area of local Voronoi areas, and \mathbf{W} is a weighted adjacency matrix, the coefficients w_{ij} of an edge being the mean of the cotangent of its two opposite angles. The geometric correspondences of these coefficients are described in Figure 5.6.

The spectral clustering requires to perform an eigen decomposition of the matrix \mathbf{L} . Let $\mathbf{U} \in \mathbb{R}^{n \times k}$ be the matrix formed with the k first eigenvectors of \mathbf{L} . Then, a spectral clustering with k clusters is obtained by performing a k-means algorithm on the rows of the matrix \mathbf{U} .

In this work, the Laplacian is computed with the python library robust-laplacian [185], and the k-means is performed with scikit-learn [157]. Once the mesh is segmented thanks to spectral clustering, each cluster can be minimally described by twelve parameters:

- three parameters to describe its position;
- nine to describe its shape and orientation.

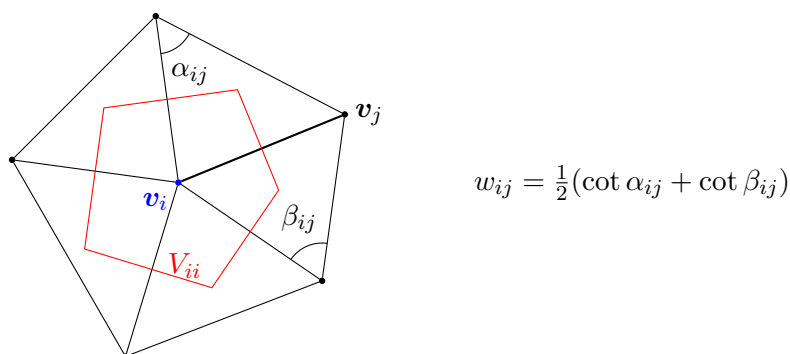


Figure 5.6 – Scheme of the one-ring triangles of a vertex v_i of a mesh. V_{ii} is the area of the Voronoi area drawn in red.

From that, many cluster descriptions can be derived, but in this work two of them have been investigated.

- The cluster approximations can be made in a Gaussian way, their positions being approximated by the means of their datapoints, and their shapes and orientations by the covariances of their datapoints. It comes down to assume the clusters are ellipsoids, and that the object can be defined as a set of such ellipsoids.
- Alternatively, it can be assumed that the object can be defined as a set of cuboids. Each cluster can be approximated by a bounding box aligned with the principal components of the datapoints. The position of the cluster is approximated by the bounding box center. Its shape and orientation is approximated by three vectors aligned with the principal components of the datapoints, that is the bounding box axes, and having their norms equal to the bounding box dimension along each of its axis.

Some examples of objects from the dataset ShapeNetCore on which the Laplacian clustering has been applied with six clusters is displayed in Figure 5.7. The two methods for cluster description (ellipsoid and cuboid) are also performed and can be compared. The Laplacian clustering is able to find a suitable and sensible object segmentation: for the mug, the different parts of the handle are separated from the cup, for the monitor, the base is differentiated from the screen, the different areas of the vase are correctly separated, and the three piping of the light bulb are in separate clusters.

However, some undesirable behaviors are also visible. First, the relative sizes of the clusters can vary substantially. For example, the power button of the monitor is identified by the Laplacian clustering as an individual cluster (in orange), while it represents a very small area and is constituted of only a few vertices. Likewise on the light bulb, a small area on its base is identified as an individual cluster (in yellow), whereas it could have been fused in the wider cluster representing the light bulb base (in purple). Using a cluster to represent such small areas represents a loss of information, as less clusters are available to encode the vast majority of the object geometry information. Then, the

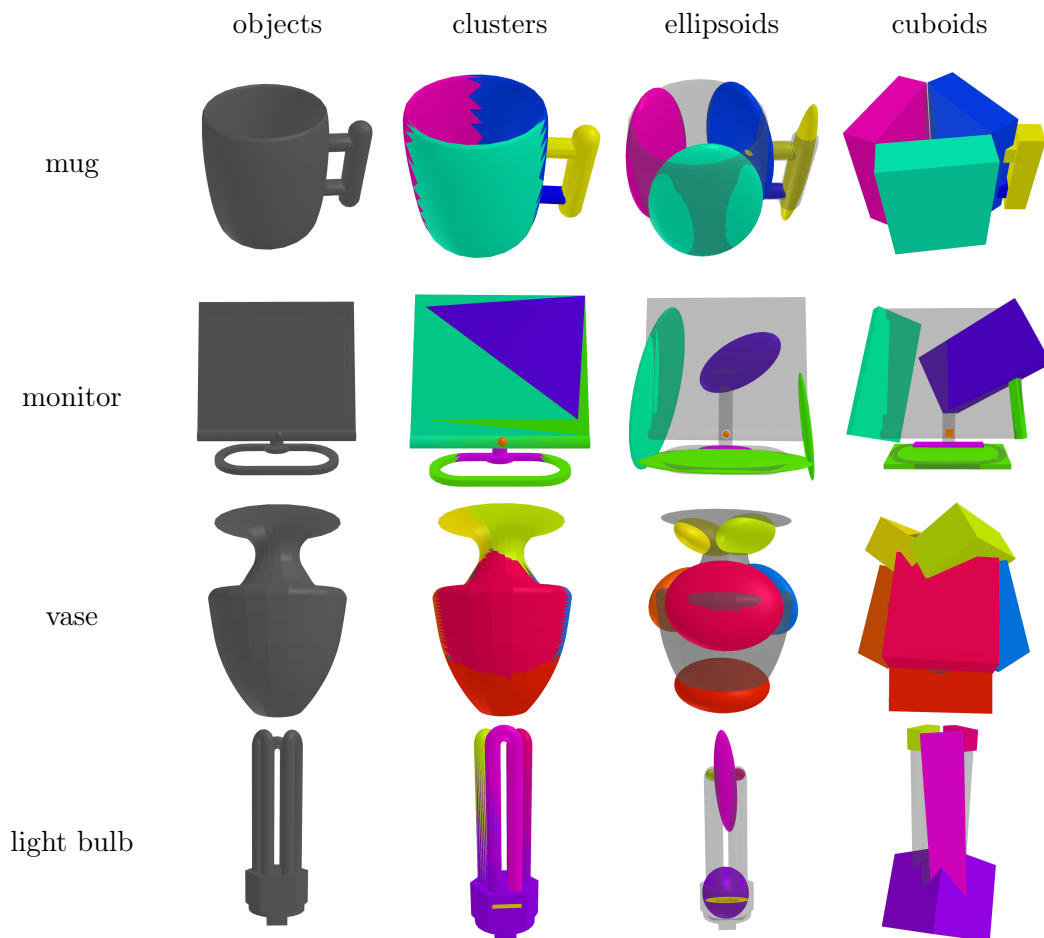


Figure 5.7 – Four object from the ShapeNetCore dataset, with their spectral clustering and corresponding approximation with ellipsoids and cuboids.

produced clusters do not preserve the existing symmetries in the object geometry. It is more visible on the ellipsoid and cuboid produced from the clusters. For example, for the monitor, it is clear that the clusters representing the screen do not behave well, as the produced cuboid or ellipsoid do not represent properly the screen geometry. Similarly, on the light bulb, the three pipings are not represented by equivalent clusters, while their geometry is identical. Due to these drawbacks of the Laplacian clustering, the produced cluster approximations does not represent faithfully the object geometry, whether cuboid or ellipsoid.

Hence, even if the geometry autoencoder would have been able to successfully learn to compress efficiently these mesh representations, it would not be very helpful as these mesh representations are not faithful to the true geometry of the object.

Vertices Sampling

Thus, although at first sight promising, the use of spectral clustering combined with cuboid or ellipsoid approximations is not satisfactory to build a constant size and compact object representation. An alternative preprocessing is to sample a fixed number of vertices from the mesh, and use the list of their position, normal vector, and first fifteen components of the Laplacian as input of the geometry autoencoder. In this case, the number of features of the geometry encoder input is $n = (3 + 3 + 15)$. The Laplacian may help to reconstruct the normal and position of the vertices. Here, to keep the feature vector as small as possible, it is chosen to sample uniformly on the mesh a hundred vertices. Thus, for a given mesh, the feature vector after this preprocessing is of dimension $n \times 100 = (3 + 3 + 15) \times 100 = 2100$. The main drawback of this approach is that it loses the graph structure of the mesh, and instead treats it as a point cloud.

As the previous attempt, the considered meshes are from the ShapeNetCore dataset. Only a subset of 11 502 meshes is considered. This subset is obtained by removing low quality meshes, for example having too many connected components, too few vertices, or ill defined vertex normal. To augment this dataset, some geometric transformations (shearing and isotropic/anisotropic scaling) are applied randomly to each mesh. Some example of such deformations are shown in Figure 5.8. As the mesh representation learned by the geometry autoencoder should also take into account the position of the object frame, the dataset is further augmented with mesh translations and rotations. This data augmentation allows to increase the number of meshes ten times, to 115 020 meshes. Then, the final dataset for the geometry autoencoder is constituted by applying twenty times the sampling preprocessing on each mesh, for a final dataset having 2 300 400 training samples.



Figure 5.8 – Example of deformations applied to the mesh to constitute the dataset: here an anisotropic scaling and a shearing is applied on both mesh.

The geometry encoder and decoder have symmetrical architectures, and are constituted of several fully connected layers with leaky Relu [186] activation. The latent space dimension is set to 64, to stay relatively close to the latent space dimension used in works dealing with geometry generation [163, 165, 166, 169, 172–174]. The architecture is displayed in Figure 5.9. The loss function is the mean squared error (MSE) between the true and reconstructed feature of each input point. Regarding the normalization, as spatial coordinates share the same unit, they are on the same scale. The same reasoning hold for normal components and Laplacian components. Thus, a normalization on the maximum and minimum value feature by feature is not mandatory. Moreover, such normalization would destroy some information about the relative point positions and normal orientations in a given sample. Thus, instead, vertex positions and Laplacian components are normalized by the global minimum and maximum of all vertex positions and Laplacian respectively. Regarding vertex normals, as they are already unit vectors, they do not require normalisation. Before the learning process, the dataset is split between a training set and a validation set. The later is used to reduce the learning rate automatically if its loss function does not improve, and stop the training if this situation last for too long. The training is performed with the RMSProp optimizer implemented in keras.

However, the training has proven to be unsuccessful, with either an underfitting problem, or an exploding gradient problem. For a relatively small model with $d = 3$ (depth parameter in Figure 5.9), the learning and validation losses stop improving after a few dozen epochs and the reconstruction performances are very poor, with a mean reconstruction error of several dozen of cm for the vertex positions and an orientation error of 85 degrees for the vertex normals, when evaluated on a test set, different from the training and validation set. Given that the objects fit in a cube of around 40 cm, and that the maximum possible orientation error of the vertex normals is 180 degrees, these numbers are very high. For larger models for example with $d = 6$, the learning encounters an exploding gradient issue after a few epochs, that is the loss value climbing to infinity. Using regularization methods such as dropout or gradient clipping failed to mitigate the exploding gradient problem. Some training tests are also conducted with different input vectors: only vertex positions, or only normals ($n = 3$ in Figure 5.9). However, the results are similar, and the network fails to learn properly. Using only the vertex positions, with $d = 3$, the mean reconstruction error reaches 8 cm, which is an improvement compared to the training using the whole feature vector, but is still far from satisfactory. Using only the vertex normals, the orientation error is still around 85 degrees. Thus, it seems that the vertex normal is particularly difficult to learn.

It is first suspected that the dataset is too complex, with too many different meshes. To test this hypothesis, a new dataset is built, with only six objects: the three objects used in chapter 4, and three other objects. The objects chosen for this restricted dataset are displayed in Figure 5.10.

Before the training, this dataset is augmented the same way as the first one based on ShapeNet, with random mesh deformations, translations and rotations, with 16 000 transformations applied on each mesh, allowing to obtain a dataset of 96 006 meshes.

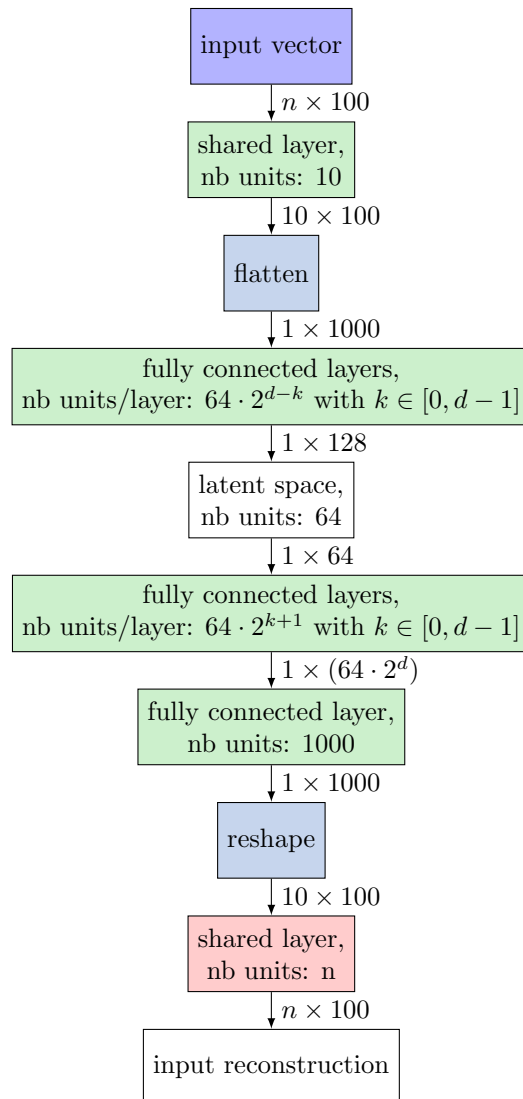


Figure 5.9 – Architecture of the geometry autoencoder trained on the dataset based on vertices sampling. In dark blue the input layer, in light blue the processing layers with no trainable weights, in green the hidden layers, and in red the output layer. d is the depth of the encoder and the decoder (that is the number of fully connected layers), and n is the number of features of the input vector.

Then, the sampling preprocessing is applied twenty times on each element to constitute the final dataset for the geometry autoencoder, the final dataset having 1 920 120 training samples.

The same architecture as previously is trained in the same way with this new dataset. However, once again the training is not conclusive, and reach a plateau after a few dozen of epoch, or encounters the exploding gradient issue for a large network. In particular,

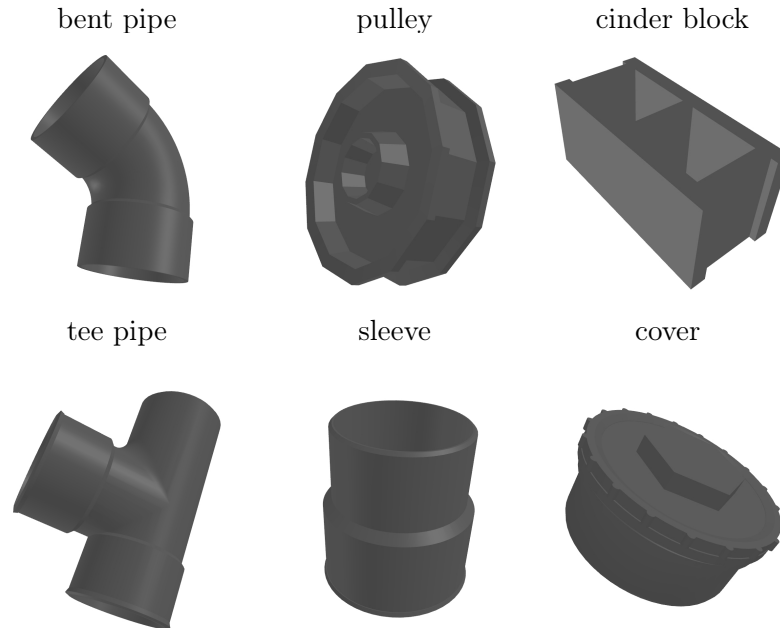


Figure 5.10 – Object chosen for the restricted dataset, to test if the failed learning is due to a too great complexity of the previous dataset based on ShapeNet.

with $d = 3$ a training test performed with an input vector using only vertex positions reaches a mean reconstruction error of 5 cm, a small improvement compared to the performances of the previous dataset, but still insufficient. On the contrary, a training test using only the vertex normal shows no improvement, with an orientation error still above 80 degrees. This confirms that the information contained in the vertex normal seems very difficult to extract and compress. From this point, the hypothesis is that the sampling is disrupting the learning. Indeed, usually, when training a neural network, each feature of the input vector has a fixed position. This position is often arbitrary, but does not change between samples, which allows the network to capture correlations existing between features. Here, the direct consequence of the vertices sampling is that the feature values can vary a lot across samples and in an uncorrelated way across features, to a point that the network seems to capture mostly noise from this dataset.

Object Geometry Remeshing

To avoid issues raised by the vertices sampling, an other technique is tested to reduce the object geometry representation dimension and to guarantee a fixed size of this representation. The proposed approach is to build a new tessellation of the object meshes so that every object has the same number of vertices. This replicates the assumption made in works dealing with the mesh generation topic.

To achieve this, each of the six objects previously used in the restricted dataset, shown in Figure 5.10, are remeshed by hand using Blender [187] to reduce their number

of vertices to 1 000 vertices. This number is chosen empirically, so that a faithful representation of the geometry is allowed with a reasonable number of vertices. In case the original mesh has less than a thousand vertices (such as the cinder block), the number of vertices is increased to reach one thousand. This remeshing process may probably be automated, but one need to keep in mind the following elements. First, the function provided by Blender acts on the number of faces, and not directly on the number of vertices. Then, the ratio between the number of faces and the number of vertices is not fixed and depends on the mesh topology. In particular, the presence of holes or handles changes this ratio.

Once all meshes have the same number of vertices, the list of their vertex positions is used for the geometry autoencoder input. As on the previous dataset based on vertices sampling, the use of Laplacian components and normals did not show any improvement on the reconstruction, they were dropped to simplify the input vector. Thus, this feature vector is of dimension $n \times 1000 = 3 \times 1000 = 3000$. The drawback of this preprocessing is the same as for the vertices sampling, that is the graph structure of the mesh is lost and the geometry is treated as a point cloud instead.

This last preprocessing technique allows to reach satisfactory compression performances with the geometry autoencoders. The conducted experiments and training trials are described in the following.

5.2 Learning to Compress Object Geometry

In this section, the methodology used to learn a compressed representation of an object geometry from the dataset based on object geometry remeshing is described, along with the obtained results.

5.2.1 Network Architecture and Training

As for the previous datasets, this new dataset based on object geometry remeshing is augmented with 35 000 random deformations, translations and rotations applied on each mesh, creating a dataset of 210 006 meshes.

The network architecture used is close to the first architecture used with the vertices sampling datasets. The main difference is a simplification of the structure by removing the shared layer at the top and final layers. Its architecture is displayed in Figure 5.11. As for the previous network, the activation functions are leaky relu, and the loss function is the mean squared error between the input and the reconstruction. In a similar way to the previous dataset, the vertex positions are normalized by the global minimum and maximum of all vertex positions. The dataset is also split between a training set and a validation set. With this new dataset, the learning is successful and does not get stuck early on a plateau of the loss function. Several architecture sizes are tested and compared: a latent space dimension l from 16 to 128 and a depth d from 3 to 5. At first, the width factor w is kept equal to 1.

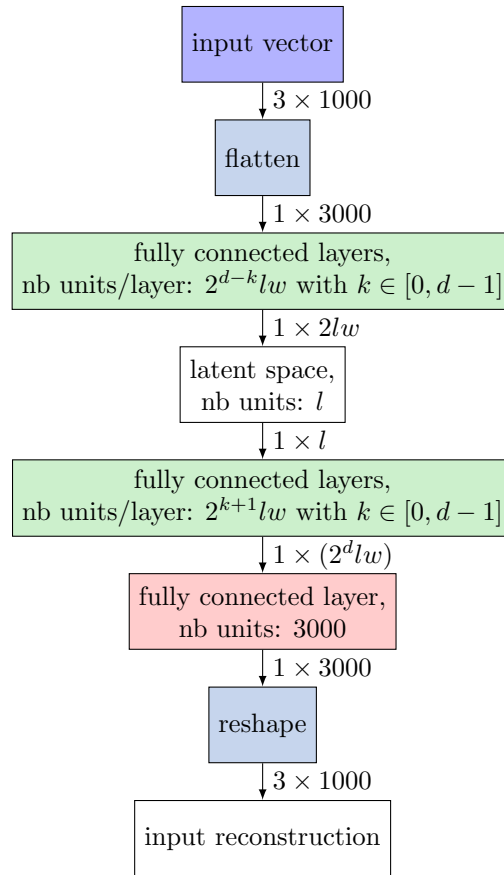


Figure 5.11 – Architecture of the geometry autoencoder trained on the dataset based on object geometry remeshing. In dark blue the input layer, in light blue the processing layers with no trainable weights, in green the hidden layers, and in red the output layer. d is the depth of the encoder and the decoder (that is the number of fully connected layers), l is the latent space dimension, and w is a width factor of the fully connected layers of the encoder and decoder.

5.2.2 Results

The conducted learning trials show that very large networks, particularly with a latent space with 128 dimensions, are more prone to the exploding gradient problem. The best reconstruction performances, evaluated on a test dataset different from the validation and training dataset, is obtained with an architecture having a latent space of 32 dimensions, a depth $d = 3$, and a width factor $w = 1$. This architecture allows to reach a mean reconstruction error of 8 mm. In Figure 5.12 is displayed some examples of meshes reconstructed by this model, compared to the given inputs. These results are exploitable, but still require some improvements. The ideal for a near perfect object geometry reconstruction would be a mean reconstruction error around 1 mm. Moreover, a latent space

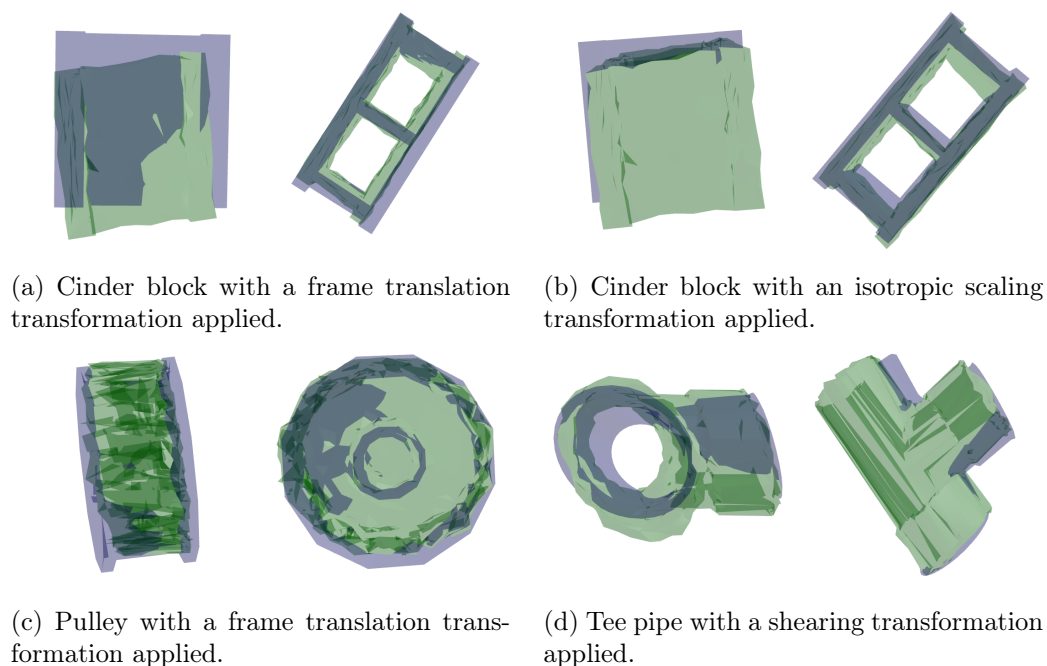


Figure 5.12 – Comparison between the reconstructed meshes (green) and input meshes (blue) for a model with $l = 32$, $d = 3$ and $w = 1$. The applied transformations on the meshes depicted here are different from the one presents in the training, validation and test sets.

having 32 dimensions is still quite large to be used in the mesh-QGG. A reduction of the latent space dimension to 16 would be ideal. Reducing further than this the latent space dimension would lead to overfitting: indeed, the dataset augmentation already implies the use of 12 variables to encode translation, rotation, shearing, and isotropic/anisotropic scaling.

The reconstruction loss on the training and validation set of this model during its training is displayed in Figure 5.13. It is clear that both curves are quite noisy, and particularly the validation one. The abrupt loss diminution at epoch 165 is due to the automatic learning rate reduction when the loss function reaches a plateau. The noise in the loss function value during training may indicate that the loss topology is chaotic, with a lot of bumps and hollows. Thus, the default learning rate may be too high, as the gradient descent bounces up and down on the loss function relief. Moreover, automatically decreasing the learning rate when the validation loss does not improve may not be optimal with a noisy loss value. Indeed, the loss function may randomly reach a very low value at a given epoch, which eventually requires a lot more epochs to be exceeded, without the cost function necessarily being on a plateau. Moreover, the abruptness of the loss decrease at epoch 165 may indicate that the automatic learning rate decrease is too steep (by default, a tenfold division) and lead the gradient descent algorithm to fall in the nearest sub-optimal local minimum.

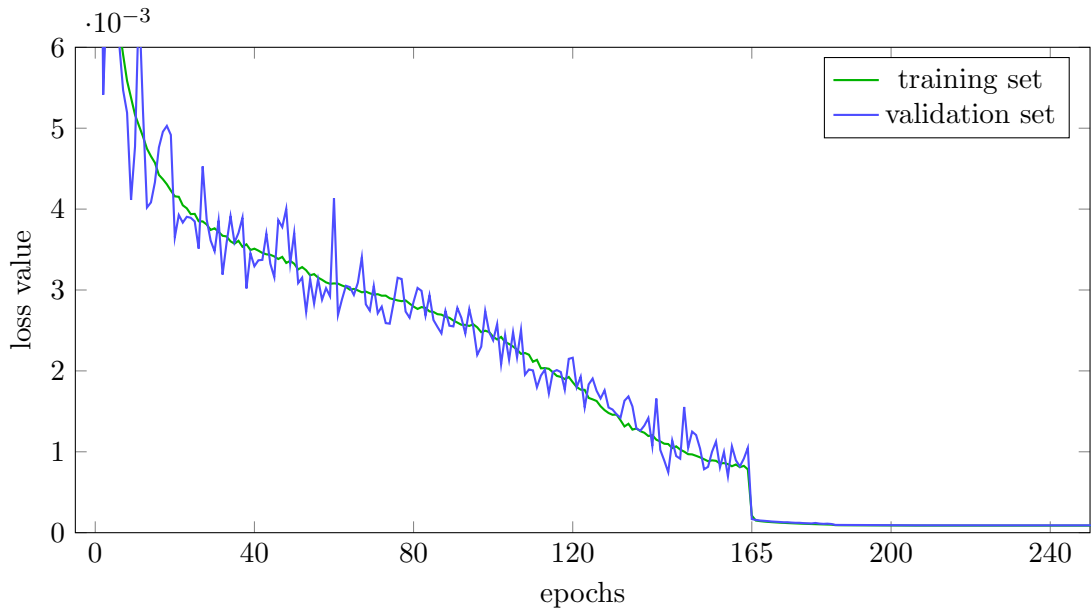


Figure 5.13 – Loss value obtained during the learning process of the geometry autoencoder, with a latent space dimension $l = 32$, a depth $d = 3$, and a width factor $w = 1$.

To mitigate these issues, further training tests are conducted with an initial learning rate divided by two compared to the default one, and the deactivation of its automatic decrease if the loss stop improving. Instead, a learning rate exponential decay is setup, with a rate of 0.9 and a step of 200 epochs: every 200 epochs, the learning rate is decreased by ten percents. The dataset augmentation procedure is also adjusted to reduce the variance in the dataset: the maximum amplitude of translation is limited to 30 cm along each axis compared to 80 cm previously, and the maximum amplitude of the scaling factor is kept inside the range $2/3$ to $3/2$, compared to the previous range $1/2$ to 2. Finally, the chaotic topology of the loss function may be mitigated by increasing the width of the network: indeed, the wider a network architecture is, that is the more units each of its layer has, the more convex its loss function is according to Li et al. [188].

The new tests are focused on architectures with a latent space with 16 dimensions. The best performances with these new settings are obtained with a depth $d = 3$ or $d = 4$ and a width factor $w = 2$ (the two networks having almost identical performances). These architectures reach a mean reconstruction error of 0.1 mm, and a median reconstruction error of 0.08 mm ($d = 3$) and 0.09 mm ($d = 4$). The reconstruction loss during training is displayed in Figure 5.14 for the architecture with $d = 3$. The graph shows that these new training settings allow to reduce the noise on the loss value during training. However, the noise is still present: it may indicate that the learning rate could be further reduced, but given the obtained performance, it seems unnecessary. The convergence is also faster, whereas the learning rate is smaller, which is counter-intuitive. This is most probably an effect of the smaller latent space: even with an increased width ratio, this architecture

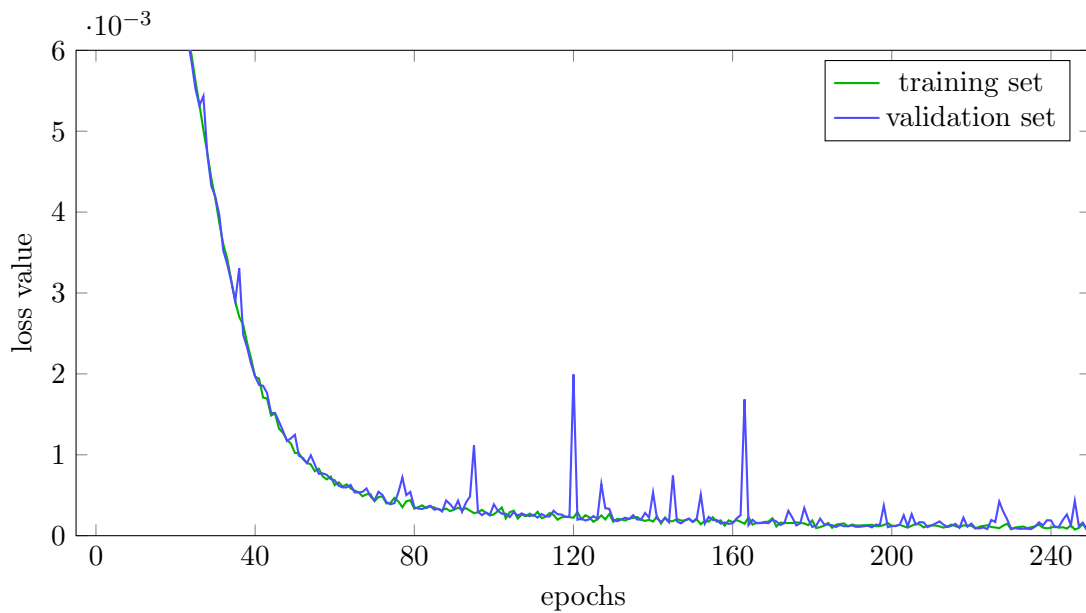


Figure 5.14 – Loss value obtained during the learning process of the geometry autoencoder, with a latent space dimension $l = 16$, a depth $d = 3$, and a width factor $w = 2$. Compared to the training graph shown in Figure 5.13, this training is obtained with an initial learning rate divided by a factor 2 relative to the default one, an exponential learning rate decay, and a dataset augmentation with reduced variance on translation and scaling.

has less trainable parameters than the previous architecture which has a larger latent space, and small networks generally converge faster than larger ones. The learning rate exponential decay also allows a steady improvement of the loss value, as the gradient descent algorithm is gradually allowed to settle on the most attractive local minimum, and cannot fall abruptly in a sub-optimal one as previously.

The reconstruction performance of this model is satisfactory (under 1 mm), and the dimension of its latent space is compatible with its use in the geom-QGG. Another aspect to consider is the regularity of the produced latent space. Indeed, if the compressed representation of different geometries are spread a lot and in a non-homogeneous way in the latent space, it could be difficult for the downstream mesh-QGG to extract the relevant information. Thus, the regularity of the latent space produced by the geometry autoencoder is important. This criterion can be quantified and compared across models by observing the non-zero standard deviations of the latent variable values computed across a given input dataset. Indeed, the smaller the mean of the standard deviations μ_σ is and the closer to zero the standard deviation of the standard deviations σ_σ is, the more homogeneous the latent space is. For example, for this model with $l = 16$, $d = 3$, $w = 2$, one obtains $\mu_\sigma = 3.40$ and $\sigma_\sigma = 0.43$.

To achieve a better regularity, some tests are conducted with the network converted

to a VAE, by adding a KL divergence term to the loss function. The goal is to keep the KL divergence loss coefficient as small as possible so that it does not increase too much the reconstruction error, but still sufficiently high to have a visible impact on the latent space regularity. A good trade-off is obtained for example for a latent space of 16 dimensions, a depth $d = 4$ and a width factor $w = 2$ with a KL divergence loss coefficient of 10^{-7} : the mean and median reconstruction error are 0.4 mm, and the regularity metrics values are $\mu_\sigma = 0.78$ and $\sigma_\sigma = 0.11$. Thus, a great improvement of the latent space regularity is obtained with a limited impact on the reconstruction. Some examples of meshes reconstructed by this model compared to the given inputs are displayed in Figure 5.15.

To further improve the regularity, some attempts to replicate the promising results presented in Burgess et al. [146] have been made. In this approach, the authors propose to start the learning with a very strong KL divergence constraint, thus drastically reducing the network capacity, and then gradually increase this capacity during the training process. Concretely, the capacity is increased by enlarging during the learning a penalty term added to the KL divergence loss component. In our tests, several amplitudes and paces of capacity increases have been tested. However, in all of our tests, although decreasing with the increase of capacity, the reconstruction loss value stay at a very high

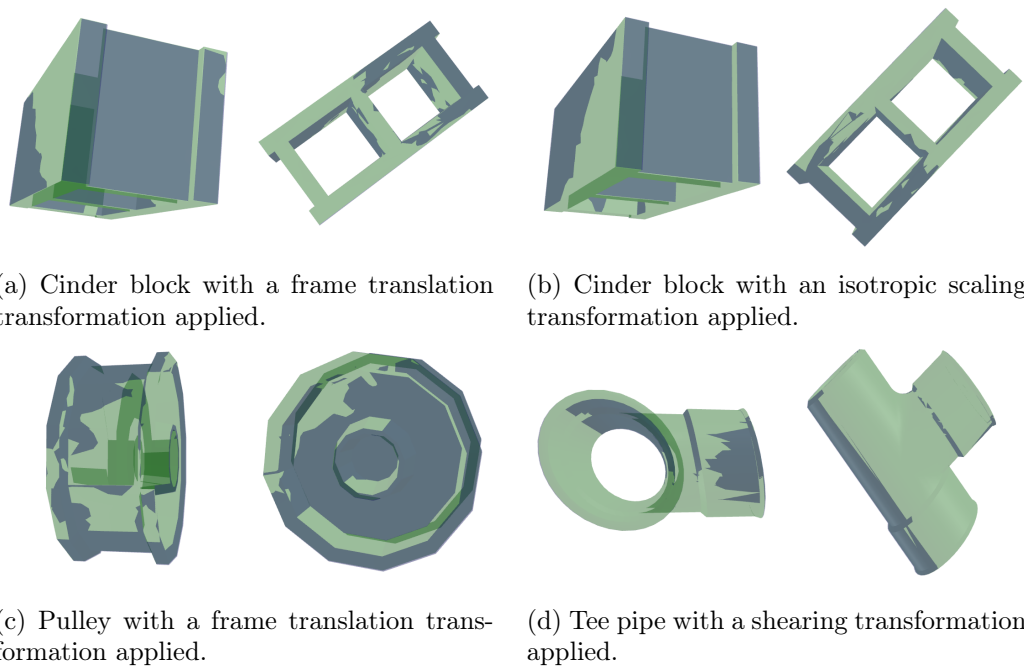


Figure 5.15 – Comparison between the reconstructed meshes (green) and input meshes (blue) for a model with $l = 16$, $d = 4$ and $w = 2$, and a KL divergence loss coefficient of 10^{-7} . The applied transformations on the meshes depicted here are different from the one presents in the training, validation and test sets.

value, orders of magnitude higher than loss values obtained with our other training. This difference may be due to our dataset that may be more difficult to learn compared to the datasets used for the tests presented in Burgess et al. [146].

5.3 Using Object Geometry to Generate Grasps

In the following, the learnt compressed representation of the geometry is used as input to train a new version of the QGG, called geom-QGG.

5.3.1 Geom-QGG Architecture and training

The geom-QGG neural network has the same global architecture as the QGG described in chapter 4, with as a new conditional input the latent variables of the geometry autoencoder. This architecture is shown in Figure 5.16. Concretely, the encoder parts of the geometry autoencoder is placed on top of the geom-QGG, with its trainable parameters frozen so that the gradient will not back-propagate through them during training. The value of its latent variables is used as input of the geom-QGG. As the geometry autoencoder is variational, its latent variable values are the result of a conditional sampling ($P(\mathbf{z}|\mathbf{x})$), which has a regularizing effect on the geom-QGG, as the geometry representation of a given object varies slightly across iterations. The training dataset is constituted by gathering in a single dataset the three grasp datasets corresponding to the three objects (bent pipe, cinder block, pulley, displayed in Figure 4.5) on which three QGGs have been trained in chapter 4.

Several training trials are conducted, with a KL loss coefficient between 4 and $5 \cdot 10^{-4}$, a number of trainable parameters between 30 000 (the network size of the original QGG) and 6 700 000, obtained by changing the depth and width of the layers inside the main encoder and main decoder in the architecture shown in Figure 5.16. Indeed, a capacity increase is probably required as the function to approximate is more complex: the dataset includes several objects and a representation of their geometries, contrary to the initial setup. To improve the learning process, a learning rate decay is added, in a similar way to the one used for the training of the geometry autoencoder. The latent space dimension is kept constant to 3, that is the same as the QGG in chapter 4.

The conducted tests show that increasing the number of parameters of the model above 930 000 does not lead to any significant performances improvements of the reconstruction performances. Regarding the KL divergence loss coefficient, the admissible range for decent reconstruction performances seems to be the same as for the first version of QGG, that is the range 10^{-3} to 10^{-4} . Tests conducted with high coefficient (1 and above), lead in the same way to very poor reconstruction, with around 3 cm of position error and around 85° of orientation error. Here, a good trade-off between reconstruction and regularity can be obtained with a coefficient of $2 \cdot 10^{-3}$.

Some comparison are also conducted between hyperbolic tangent and leaky relu activation function, to verify the influence of the network capacity on the performances with these two functions. Indeed, hyperbolic tangent is often the default choice, but the relu

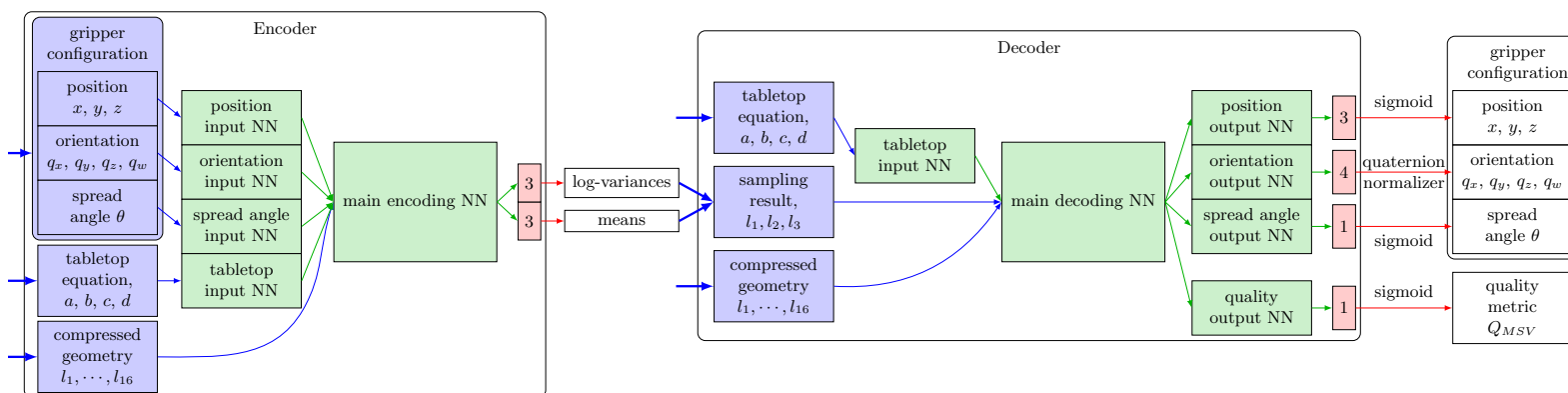


Figure 5.16 – Geom-QGG architecture. In blue the input layers, in green the hidden Neural Network (NN) and in red the output layers. The hidden NN inner layers are fully connected layers, with hyperbolic tangent or leaky relu activation functions. The main encoding and main decoding NN have symmetrical inner architecture. The inputs and outputs NN are small networks (with much less parameters compared to main encoding and decoding NN) in charge of extracting or reconstructing specific features associated with position, orientation, spread angle, tabletop equation or quality respectively. The supplementary input for the tabletop plane Cartesian equation and compressed geometry ensures that the generated grasp depends on them [147]. The tabletop input NN appears both in encoder and decoder: it is the same network in both places (same weights), and not two different networks. The compressed geometry is obtained through the latent space of the geometry encoder. This architecture is implemented with Tensorflow [156] and Keras [155] python libraries.

and leaky relu functions are more suited for larger and deeper architectures. These experiments show that for a model with 3 500 000 parameters, the reconstruction performances with hyperbolic tangent are significantly worse than for a model with 930 000 parameters, by a factor between 1.5 and 5, depending on the reconstructed feature and on the used KL divergence loss coefficient. On the contrary, with the leaky relu, the reconstruction performances does not change between the two model sizes, and are identical to the one reached with hyperbolic tangent for the model with 930 000 parameters.

Thus, after these various tests, a network having 930 000 trainable parameters, a KL divergence loss coefficient of $2 \cdot 10^{-3}$, and using the hyperbolic tangent activation function is selected. This architecture reaches a mean position reconstruction error of 0.005 m and a mean orientation reconstruction error of 3.51 degrees.

5.3.2 Experiments

First, the grasp planning procedure presented in subsection 4.3.2 is also applied in simulation on the three objects used for the geom-QGG training. The grasp planning used is described in Algorithm 1. As in subsection 4.3.2, the procedure is repeated on 1000 random object poses for each stable position for each object. In the same manner, three metrics are monitored:

1. the grasp success rate,
2. the number of collision and reachability checking iterations needed to find three admissible grasps (Algorithm 1 line 5),
3. the grasp quality prediction relative error.

These metrics are summarized in Table 5.1.

	1) success rate (%)	2) Algorithm 1 line 5 mean iterations	3) mean quality prediction error (%)
bent pipe	99.8	6.4	23.9
cinder block	98.2	5.2	18.2
pulley	99.3	5.3	25.5

Table 5.1 – Performances of the geom-QGG on simulated grasp planning trials.

Here, the success rates for the pulley and cinder block are still very high, but slightly lower than the ones reached with the classic QGG. The bent pipe, for its part, has a slightly higher success rate than the one achieved by the classic QGG. The mean grasp quality prediction error on its part is higher than for the original QGG for the three objects. As this evaluation highly depend on sampling, these differences can be due, at least in part, to the variance of the evaluation method. An assessment of this variance could be interesting in future works. An other part of the performance differences can of

course be explained by the fact that generating grasps and predicting quality metrics in a multi-object framework, as the geom-QGG, is inherently more complex than generating grasps and predicting their quality for a single object, as the classic QGG. Finally, the number of collision and kinematic reachability checking iterations is comparable to the ones obtained with the classic QGG.

To qualitatively assess the generalization abilities of the geom-QGG, the produced latent space is manually explored, for non-learnt objects, as well as for a deformed version of one of the learnt object.

For the non-learnt objects, the three other objects used in the geometry autoencoder dataset are tested: the tee pipe, the sleeve, and the cover (shown in Figure 5.10). Unfor-

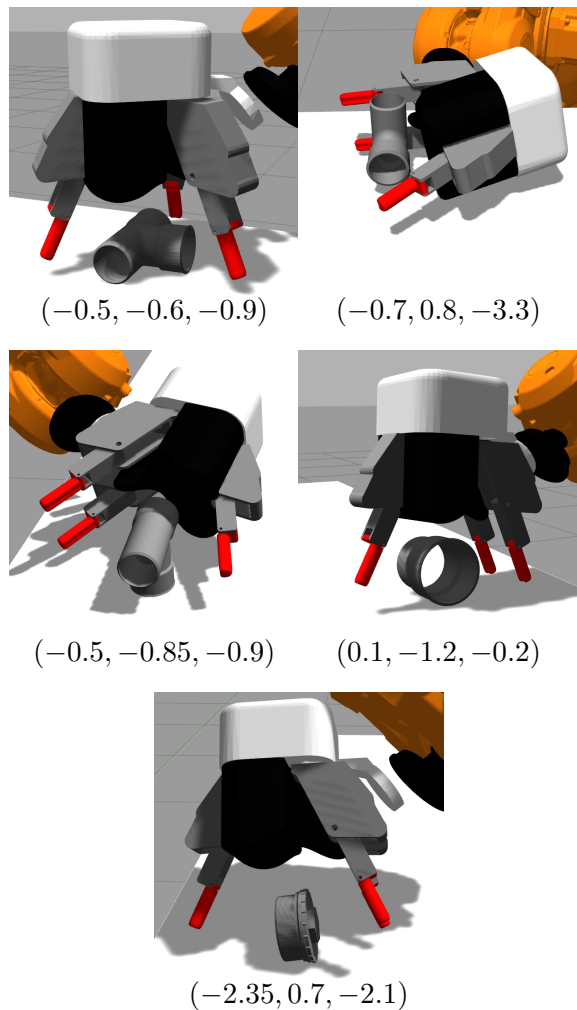


Figure 5.17 – Examples of gripper configurations with their latent space coordinates, generated for non-learnt objects in latent space areas that are likely to produce successful grasps. these areas are found by manual inspection of the latent space.

tunately, after manually inspecting the latent space, it appears that the great majority of their latent space produces inconsistent grasps. This is not surprising, as with only three learnt objects, it is extremely complex to extrapolate grasps for completely different geometries. However, it is still possible to find latent space areas that produce relevant grasps: some promising grasp configurations are displayed in Figure 5.17, along with the corresponding latent variable values. The closer the grasp is from the coordinate $(0, 0, 0)$ in latent space, the easiest it is to draw it by sampling from a normal distribution.

Regarding the deformed version of one of the learnt object, a scaled version of the cinder block is used as an example (with an isotropic scaling of 10 %). It is displayed in Figure 5.18, along with the original cinder block used during learning of the geom-QGG.

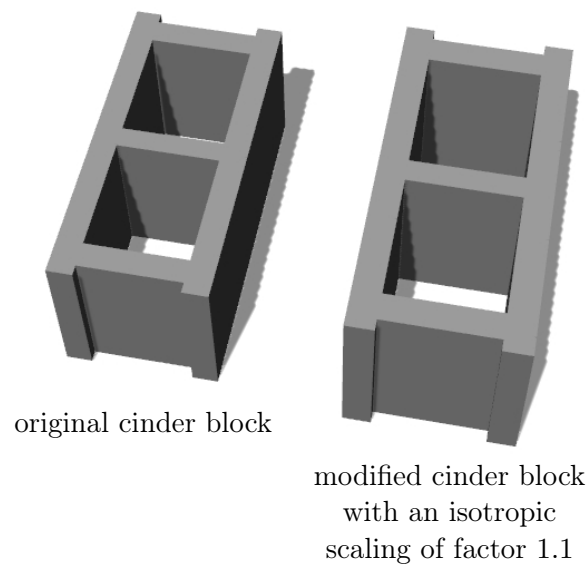


Figure 5.18 – Comparison between the original version of the cinder block, and a version modified by applying an isotropic scaling of factor 1.1.

When manually exploring the deformed cinder block latent space, it is clear that the grasps are organized in it in a very similar way to the original object. The primitive grasps specified for the original cinder block can be retrieved in the latent space of the deformed one, and often in areas near the same coordinates in latent space. This is highlighted in Table 5.2, where some grasps are shown with their coordinates in latent space for both original and deformed cinder blocks. This table is created as follows:

1. some specific grasps are identified in the latent space of the original cinder block (for example, some that are well aligned and centered relative to the object);
2. their coordinates in latent space are registered;
3. the same coordinates are used to produce a gripper configuration in the latent space of the deformed cinder block;

4. if the produced gripper configuration does not resemble the initial configuration produced for the original cinder block, the latent space of the deformed cinder block is manually explored, starting from the registered coordinates, to find the closest area that produces gripper configurations similar to the one produced for the original cinder block.

For two grasps, the same latent space coordinates produce very similar grasps for both objects. The other grasps need adjustment to the latent space coordinates to be reproduced, the greatest displacement being required by the fourth grasp, with a distance of 1.8 in latent space between the two coordinates. Thus, it is likely that for objects that resemble known primitive objects, relevant grasps may be generated, but in a less reliable way, depending on how close the object is to its original.

grasps from original cinder block	grasps from deformed cinder block	
$(0, -0.5, 0)$	$(0, -0.5, 0)$	
$(0, 0.5, 0)$	$(0, 0.5, 0)$	$(0, 1, -1)$
$(0.5, 0.5, -0.4)$	$(0.5, 0.5, -0.4)$	$(1.2, 0.7, -0.7)$

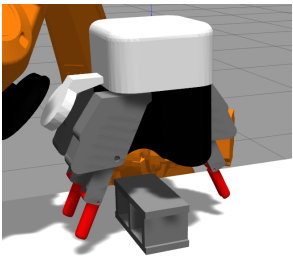
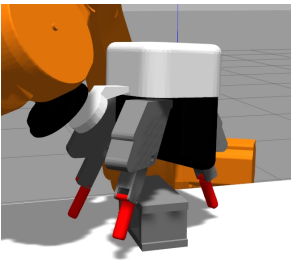
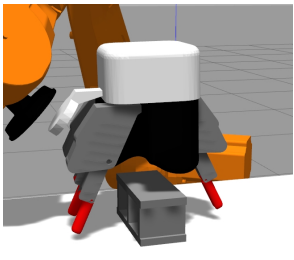
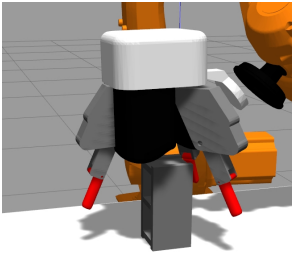
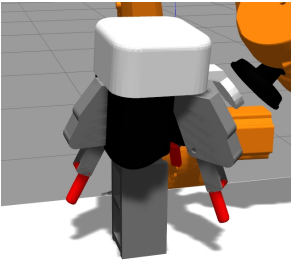
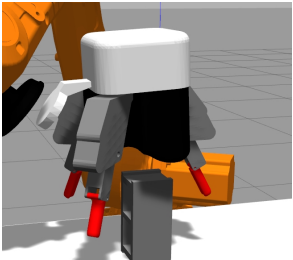
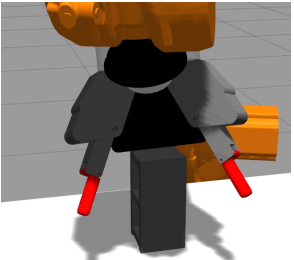
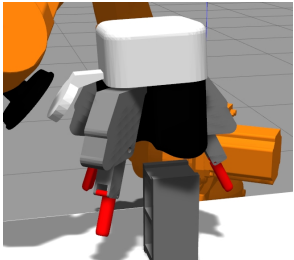
grasps from original cinder block	grasps from deformed cinder block	
		
$(0, -0.5, 0)$	$(0, -0.5, 0)$	$(0.8, -0.6, -1.6)$
		
$(1, 0.5, 0)$	$(1, 0.5, 0)$	
		
$(0, -0.5, 0)$	$(0, -0.5, 0)$	$(1.2, -0.5, 0)$

Table 5.2 – Comparison between grasps generated for the original and deformed cinder blocks. First, grasps are selected in the original cinder block latent space (left). Then, the goal is to find grasps resembling to them in the deformed cinder block latent space (right). In the middle is displayed the grasps generated for the deformed cinder block at the same latent space coordinates as the selected grasp on the left.

5.4 Conclusion

In this chapter, a variant of the QGG, called geom-QGG, has been introduced. It uses object geometry information to generate potential grasps along with the expected grasp

quality. The goal is to be able to use a single network to generate grasps for several learned objects, instead of one network per object. Experiments show that this variant allows to reach performances comparable to the original QGG implementation when evaluated on a grasp planning task on learned objects. Preliminary results also suggest that this variant may produce relevant grasps for deformed versions of the learned objects, but with a probable loss of reliability, depending on the likeness of the deformed object to the original one. For entirely different objects, the latent space is mainly filled with irrelevant grasps, but some consistent ones can still be found. Thus, further developments are still required to confirm and improve the generalization capability of this variant.

As object geometry descriptions can represent a lot of information, the geom-QGG takes as input a compressed representation of the object geometry, so that it keeps a relatively small sized input. This compressed representation is learned with an autoencoder network. This network is able to compress a geometry represented by a thousand points, to a latent space of sixteen dimensions. However, its performances needs to be confirmed on a dataset with a greater variety of geometries.

Conclusions & Perspectives

Conclusions

Robotic grasping is a research field that is very active and has promising industrial application prospects, and more particularly in the complex case of an underactuated and pluri-digital gripper architecture. Indeed, such architectures are more versatile than classic grippers currently used in industry.

However, it has been shown that the development of underactuated and pluri-digital grippers faces several challenges, especially regarding their associated grasp planning algorithms. The goal of the grasp planning algorithm is to find, for a given gripper and object, an appropriate gripper configuration, that allows to grasp the object reliably, and in a way compatible with the task. This requires to explore the space of all possible grasps for a given object-gripper configuration. In this work, this space is designated as the grasp space.

For grippers having a low dimensional configuration space, such as bi-digital parallel grippers, the grasp space exploration can be done by extensive testing of different possible gripper configurations. It is difficult to extend this approach to more versatile grippers with a lot of degrees of freedom, as the space to be explored would be too large.

For fully actuated pluri-digital grippers, the grasp space exploration can be done through exploring kinematically accessible contact positions on the object. Nevertheless, this approach cannot take into account a high number of contact points, which often limits it to precision grasps. Moreover, such grippers are costly, and the high number of degrees of freedom complexify their mechanical architecture and controller.

The constraints of adaptive and underactuated mechanisms aim at simplifying the controller and lowering the cost of pluri-digital grippers by reducing the number of controlled degrees of freedom. However, it has been shown that the position of the contact points on the object cannot be determined only based on the knowledge of actuator configurations, as the finger configurations depend on a force equilibrium between the object and the fingers.

Thus, in the underactuated case, the grasp space exploration can only be done by testing different gripper configurations, and using dynamics simulation to predict the resulting grasp configuration. It can be highly inefficient and time consuming due to the number of controlled degrees of freedom, which is higher than for simpler grippers. In the literature, there are few works that deal with the grasp planning issue for pluri-digital

and underactuated grippers. Moreover, they often make simplifying assumptions that reduce the dimension of the grasp space, which can limit the versatility of the approach.

The main contribution presented in this thesis is a human initiated and object dependant method to efficiently explore the grasp space for an underactuated and pluri-digital gripper. It is based on three main intuitions:

- human are able to intuitively find pertinent gripper configurations belonging to the grasp space;
- building a model of the grasp space allows to generate grasps belonging to it with a high probability;
- the grasp space can be modeled as a set of low-dimensional submanifolds of the gripper configuration space.

This method uses Variational Auto-Encoders (HGG and QGG) to learn a model of the grasp space from a set of human-provided grasp primitives, together with a prediction of the value of an analytic grasp quality metric. It has been demonstrated that this method allows to produce grasps both in simulation and on a real robotic setup in a very reliable way. Moreover, the proposed method has a success rate at least ten times higher than other commonly used grasp space exploration methods that are based on random sampling. Our method also produces grasps with a higher quality value on average. Finally, a variant of this method taking into account the object geometry has been proposed. Still, several improvements can be made on various aspects of the method, and some of these future research perspectives will be discussed in the following.

Perspectives

1 Intuitive Programming for Specifying Primitive Grasps

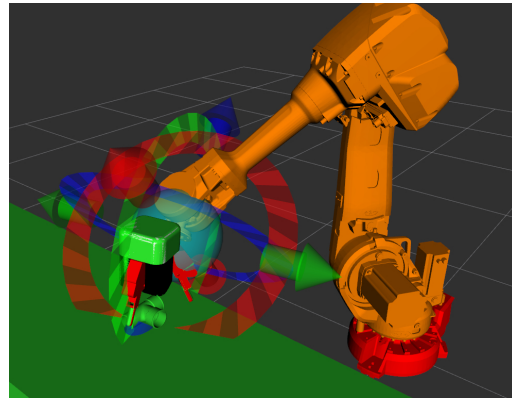
The main constraint of the proposed method is the need to provide human-defined grasp primitives. It allows to produce with a high success rate high quality grasps, however, the constitution of the primitive grasp set is time consuming, and can be tedious for the operator. To reduce the tediousness and time-consumption, it is important to give a special attention to the user interface used to specify primitive grasps, and design it carefully. It needs to be as efficient and intuitive as possible.

For example, a very intuitive way to proceed can be to use a virtual reality environment to specify the primitive grasps. The operator would be able to visualize and control very quickly and intuitively the gripper pose relative to the object, as well as its internal configuration. This possibility is shown in Figure 4.

An other possible solution is to use the comanipulation abilities of a collaborative robot. When set in gravity compensation mode, the robot only counteracts the effect of its own weight, but offer no resistance to any other forces. One can use this mode to position and orient the gripper in the desired pose by direct physical interactions. The gripper should also have an equivalent mode to set its internal configuration by moving its links relative to each other. This is illustrated in Figure 5.



(a) Example of a virtual reality setup: a headset and two controllers with motion tracking. The controllers allow to interact with the virtual environment, for example to control the desired pose of the gripper. Picture: CC BY license [189]



(b) The MoveIt Rviz plugin [154], an example of simulation environment that could be used in virtual reality to set the primitive grasps. The red, green, blue arrows and circles are designed to move the end-effector with the mouse, but the same functionality could be achieved through motion tracking.

Figure 4 – Use of virtual reality to specify primitive grasps.



Figure 5 – Use of a collaborative robot to specify primitive grasps. When in gravity compensation mode, the operator is able to move the robot arm end-effector in any desired configuration (here with a Kuka iiwa): it could be used to set the primitive grasps. Picture: copyright ©2022 KUKA.

2 Studying Ways of Reducing the Primitive Grasps Dataset

Another possible improvement related to the human-provided primitive grasps concerns the required number of primitives. Indeed, as any machine learning algorithm, being based on data, the initial HGG training requires a reasonable amount of input data,

namely the primitive grasps. Reducing the size of the training dataset will reduce the quality of the learned grasp space model. However, a reduction of the number of human-provided grasps is desirable, to reduce the time investment required by the method. Here, recall that for a given object the primitive grasps can be divided in several grasp types. Each of these grasp type can be seen as continuous and bounded area in the gripper configuration space. Thus, by providing only a few grasp primitives located on the borders of each grasp type area, it should be possible to interpolate its bounding surfaces. This interpolation requires to make some assumptions on the shape of the grasp type area in the gripper configuration space. An example in two dimensions of such approximation for a grasp type is given in Figure 6. Then, it is possible to sample new positions in the space bounded by these surfaces.

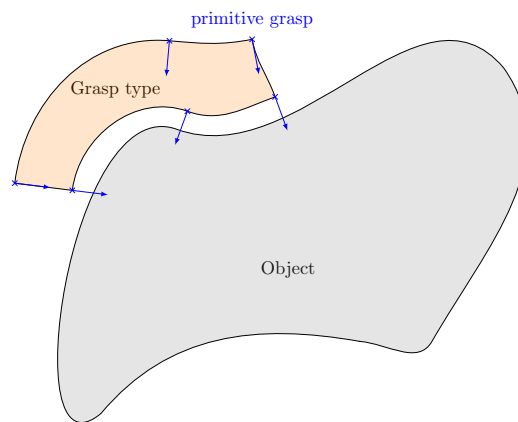


Figure 6 – Example in a two dimensional cut of the interpolation of a grasp type area from six primitive grasps, visible in blue. The blue crosses represents their positions, and the blue arrows the gripper palm orientation at these positions.

This should allow to create at a lower time cost the training dataset for the HGG: indeed, one needs to provide only a few primitive grasps, typically no more than a dozen for each grasp type, instead of around thirty for each grasp type in the current implementation. Even in this framework where a continuous representation is available for each grasp type, the HGG is still pertinent: it will be able to extract the complex relationship existing within each grasp type and between them to map them in a single low dimensional representation.

3 Towards the Consideration of Task Constraints in the Methodology

The presented grasp space exploration method focuses on the stability of the grasp, and do not consider other aspects, such as the compatibility of the selected grasp with the task requirements. Taking into account the task requirements into the grasp planning

is complex, but also closer to a real use case. In the literature, mainly two approaches exist.

- One aims at extracting semantic information from the object geometry to produce grasps on areas specific to a task, for example identifying a handle, so that grasps can be performed on it [190, 191].
- The other approach expresses the task as a set of disturbing wrenches, and design metrics measuring how well a grasp is able to resist them [192, 193].

Regarding the first approach, specifying primitive grasps partially fills this function: indeed, if the object has a part of its geometry clearly dedicated to grasping, the human operator will very likely specify primitive grasps on this area.

Regarding the second approach, provided that the task is expressed in the object frame, a task requirement metric could be used instead of the grasp quality metric, or a fusion of both. However, if the task can be expressed only in a fixed reference frame, different from the object frame, our method cannot directly take it into account, as the generated grasps are expressed in the object frame. A supplementary step can still be added to check the compatibility of the generated grasp with the task requirements, in the same way as the kinematic reachability check.

Special Case of a Pick & Place Task

In this case, it is important to know if the generated grasp is compatible with the release pose of the object, that is if there is no collision between the gripper and the tabletop in the release phase. In the framework of our method, the release pose can be modeled through the tabletop plane Cartesian equation of the corresponding stable pose \mathbf{e}_r (as the initial grasp stable pose, \mathbf{e}).

If \mathbf{e}_r is identical to \mathbf{e} , then any collision free grasp is compatible with the release stable pose. However, if \mathbf{e}_r is different, the compatibility needs to be ensured. An example of such situation is displayed in Figure 7. This could be done when testing generated grasps in simulation to constitute the QGG training dataset. If the tested release stable pose is not compatible with the tested grasp, the grasp quality metric could be set to zero for this given grasp and release stable pose. Then, the QGG could learn to generate grasps and associated quality with two conditions as input: \mathbf{e} and \mathbf{e}_r . This change to the QGG is shown in Figure 8.

4 Improvements of the Methodology Variant Based on Object Geometry Features

Regarding the proposed variant that takes into account the object geometries, several improvements can be made to the geometry autoencoder. First, it should be trained on a wider dataset, to ensure that it can acquire generalisation capabilities. Moreover, it should be able to take advantage of the progresses that will most likely be made in the upcoming years by the deep learning research community dealing with geometry

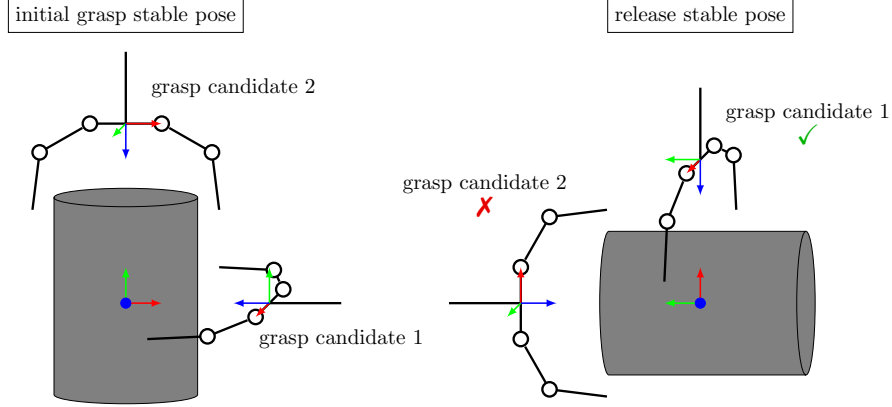


Figure 7 – Case when the initial grasp stable pose is different from the release stable pose. Here, the grasp candidate 2 is not compatible with the release stable pose due to collision with the tabletop surface.

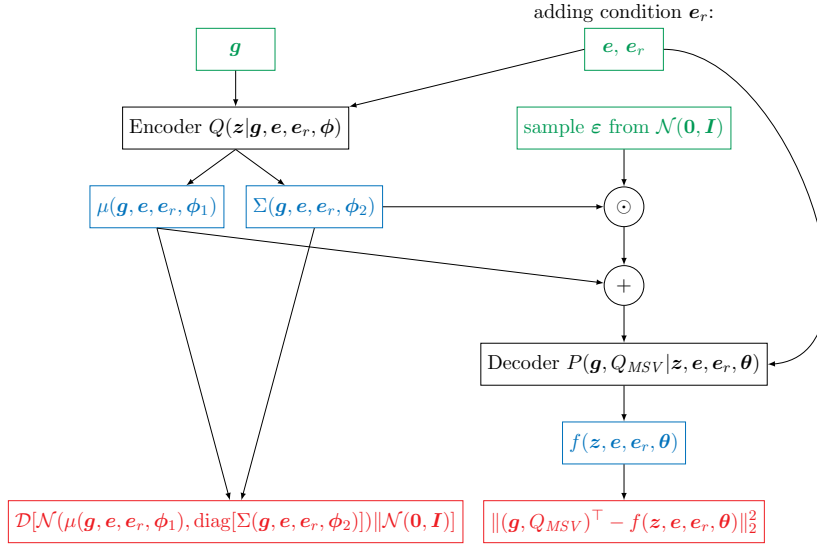


Figure 8 – Possible implementation of a QGG taking into account the object release pose. In green the inputs, in blue the outputs, and in red the loss function components. e is the initial grasp stable pose, and e_r is the additional condition corresponding to the release pose.

processing, in order to use the graph structure of mesh representation. Even for the point cloud representation currently used, some improvements can be made, especially to be invariant to point ordering. Indeed, the mean squared error loss function currently used forces the reconstruction to reproduce the point ordering of the input, whereas it does not matter in a point cloud or in a mesh. In the literature, there exist metrics allowing to compare two unordered point sets, such as for example the Chamfer distance

[194], that measures the squared distance between each point in one set to its nearest neighbor in the other set. Using such distance as loss function should allow to learn a model that does not depend on the point ordering, although it is more computationally costly than the mean squared error.

Finally, some improvements could also be made regarding the geom-QGG. The input grasp dataset could be augmented by applying rotations and translations to the object frame: this way, it would be easier for the geom-QGG to identify in the geometry autoencoder latent space which latent variables represent rotation and translation of the object frame, and which are related to the object geometry itself. To tackle the problem of unknown primitive objects, inspiration could be taken from reinforcement learning. The first step would be, after an initial geom-QGG training, to generate new grasps for a set of unknown primitive objects. Then, a second step would be to test them in simulation. If the object is not a deformed version of one of the known primitive object, a lot of the generated grasps will be failed grasps, but a small proportion will eventually succeed. Lastly, it would be possible to re-train the geom-QGG with a new dataset including these new successful grasps corresponding to unknown primitive objects. Repeating this process several times may improve the performances. Keeping a given proportion of unknown primitive object failed grasps in the new dataset may also be profitable. This way, the grasp space exploration procedure may acquire interesting generalisation capabilities.

Possible Applications

If grasps can be generated with the proposed grasp space exploration method for a sufficiently large set of objects, it is possible to generate optimal grasps that could be used as training data for a versatile grasp planner. This grasp planner would use a RGB or depth image as input for example, and output grasps in the camera frame. It will benefit from the high quality of the grasps generated by our method, compared to classic grasp space exploration methods based on gripper configuration space sampling. Provided that it would be trained on a sufficiently large image dataset, it may be able to produce grasps with unknown object geometry, and without any a priori knowledge of the object position in the scene.

The proposed method is also well suited to the framework of industry 5.0, in particular to the small batch manufacturing issue. When the objects to be handled have a complex shape, and are modified after a few dozen or hundred produced parts, the manufacturing process can be difficult to automatize. Our method can help to drastically reduce the time required for the adaptation to a new object geometry: an operator only needs to provide new primitive grasps. This way, the operators can focus on tasks with high added value, while letting a robot take care of handling tasks.

Some potential applications are also in assistive robotics. Indeed, there are already some robotic arms that are adapted for this use case, for example the Jaco arm from Kinova [195]. Our method could ease the use of such systems, by providing automated and reliable grasping abilities for a set of common household items for example. This could help to improve the autonomy and quality of life of disabled persons.

Appendix A

Details of the Experimental Setup

In Figure 1 is schematized one of the three underactuated fingers of the gripper used in chapter 4. Kinematically, the proximal joint position θ_2 and the distal joint position θ_3 are linked to the actuator position θ_m by the two following relations:

$$\theta_{2a} = \beta - \tan^{-1} \left(\frac{L_h}{L_v} \right) - \cos^{-1} \left(\frac{r(\theta_m - \theta_{m0})^2 - L_t^2 - D^2}{2L_t D} \right) \quad (\text{A.1})$$

$$\theta_{2a} = \theta_3 + \theta_2 \quad (\text{A.2})$$

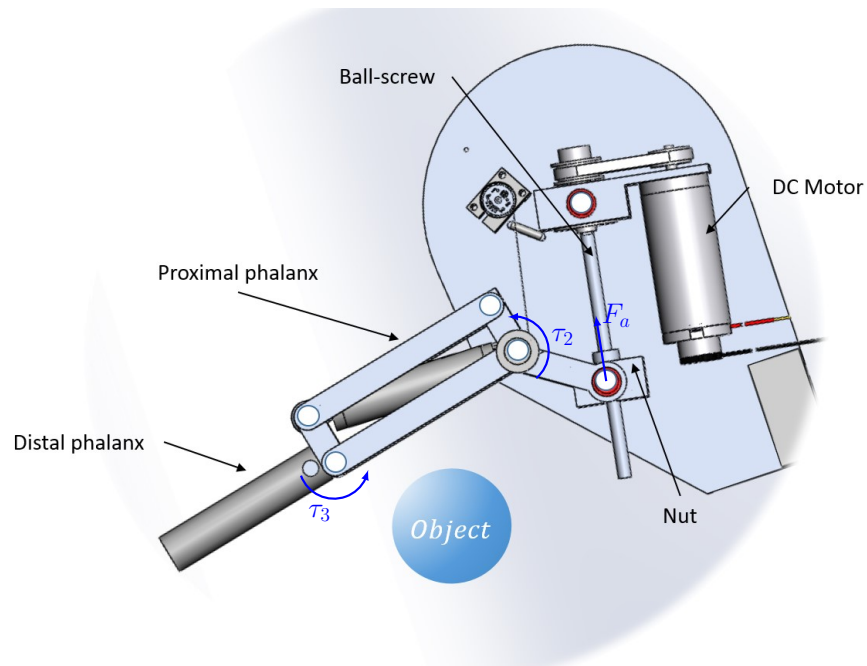
with r a reduction factor corresponding to the step of the actuation screw, the other quantities being annotated in Figure 1b. Regarding the dynamics, the proximal joint torque τ_2 and distal joint torque τ_3 are expressed as follows:

$$\tau_2 = F_a \sqrt{L_t^2 - \left(\frac{L_t^2 - D^2 + ((r(\theta_m - \theta_{m0}))^2)}{2r(\theta_m - \theta_{m0})} \right)^2} \quad (\text{A.3})$$

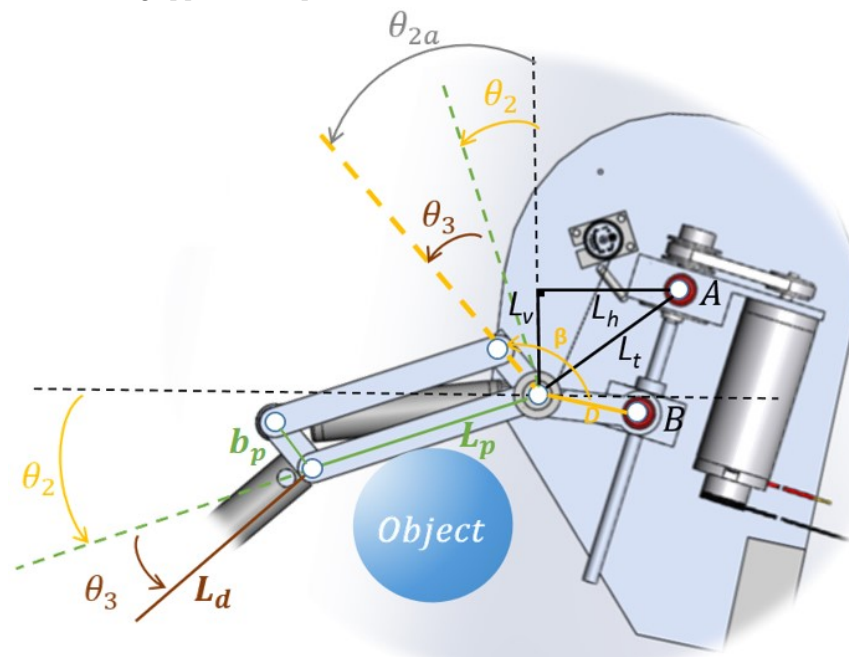
$$\tau_3 = -k \frac{b_p L_p \cos(\theta_3)}{\sqrt{(b_p \cos(\theta_3))^2 + (b_p \sin(\theta_3) + L_p)^2}} \left(\sqrt{(b_p \cos(\theta_3))^2 + (b_p \sin(\theta_3) + L_p)^2} - L_0 \right) \quad (\text{A.4})$$

with k the stiffness of the spring located in the four bar linkage mechanism in the proximal phalanx.

The general architecture of the robotic setup is displayed in Figure 2. A supervision PC running ROS is used to execute the grasp planning algorithm. This planning algorithm centralize the information from the robot, the camera and the gripper. It is in charge of generating candidate grasps and checking their reachability using Moveit [154]). It also sends commands to the robot arm and to the gripper to execute the grasp.



(a) Scheme of the elements constituting an underactuated finger of the considered gripper in chapter 4.



(b) Annotated scheme of an underactuated finger of the considered gripper in chapter 4, with quantities useful to compute joint torques and kinematic relations between joints.

Figure 1 – Scheme of a finger of the considered underactuated gripper in chapter 4.

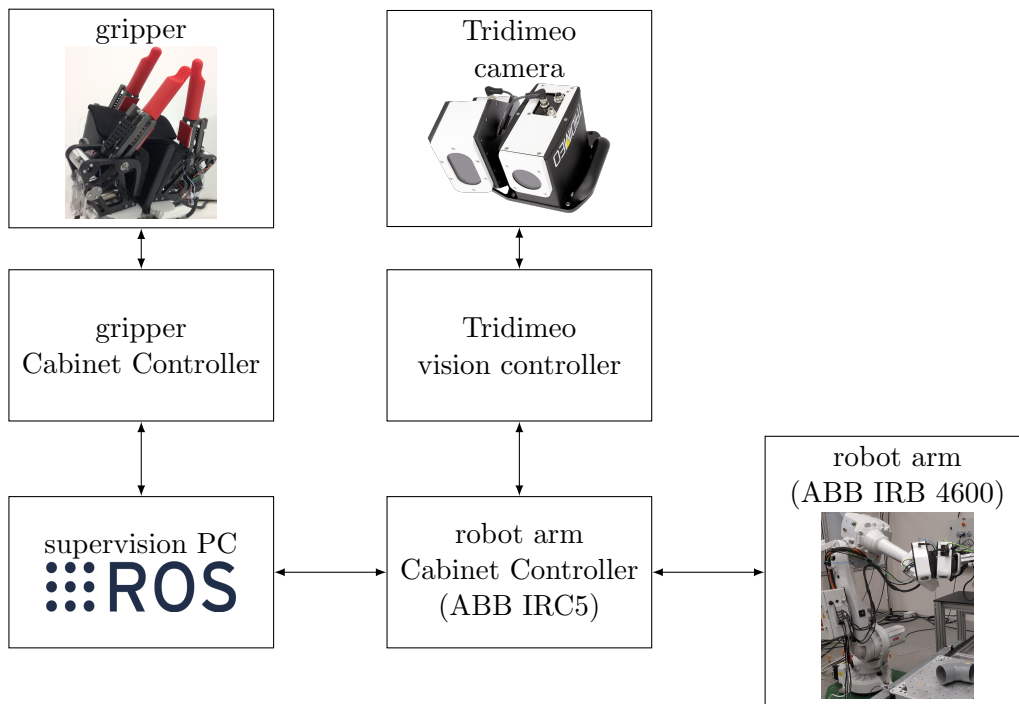


Figure 2 – Scheme of the global control architecture used on the real robotic setup.

Nomenclature

Grasp Modeling

$\tilde{\xi}_f$	total contact twist applied by a finger
$\tilde{\xi}_g$	total contact twist applied by a gripper
$\tilde{\xi}_i$	applied contact twist at contact i
\tilde{G}	Grasp Map (not taking into account contact models)
\tilde{J}_a	finger actuator Jacobian matrix (not taking into account contact models)
\tilde{J}_f	finger Jacobian matrix (not taking into account contact models)
\tilde{J}_g	gripper Jacobian matrix (not taking into account contact models)
\tilde{J}_j	finger underactuated joint Jacobian matrix (not taking into account contact models)
\tilde{J}_{ag}	gripper actuator Jacobian matrix (not taking into account contact models)
\tilde{J}_{jg}	gripper underactuated joint Jacobian matrix (not taking into account contact models)
\tilde{w}_f	total contact wrench applied by a finger
\tilde{w}_g	total contact wrench applied by a gripper
\tilde{w}_i	applied contact wrench at contact i
ξ_f	total contact twist transmitted by a finger
ξ_g	total contact twist transmitted by a gripper
ξ_i	transmitted contact twist at contact i
ξ_o	object twist
G	Grasp Map (taking into account contact models)

NOMENCLATURE

\mathbf{J}_a	finger actuator Jacobian matrix (taking into account contact models)
\mathbf{J}_f	finger Jacobian matrix (taking into account contact models)
\mathbf{J}_g	gripper Jacobian matrix (taking into account contact models)
\mathbf{J}_j	finger underactuated joint Jacobian matrix (taking into account contact models)
\mathbf{J}_{ag}	gripper actuator Jacobian matrix (taking into account contact models)
\mathbf{J}_{jg}	gripper underactuated joint Jacobian matrix (taking into account contact models)
\mathbf{T}_f	finger transmission matrix
\mathbf{T}_g	gripper transmission matrix
\mathbf{w}_f	total contact wrench transmitted by a finger
\mathbf{w}_g	total contact wrench transmitted by a gripper
\mathbf{w}_i	transmitted contact wrench at contact i
\mathbf{w}_o	object wrench
K_i	stiffness coefficient of the underactuation mechanism at joint i
n_λ	total number of twist or wrench component transmitted by the contacts generated by a finger or a gripper grasping an object
n_c	number of contacts generated by a finger or a gripper grasping an object
n_q	number of joints of a finger
$n_{\lambda i}$	number of twist or wrench component transmitted by the contact i
n_f	number of fingers of a gripper
n_{qg}	number of joints of a gripper
X_i	coefficient of the transmission matrix relating q_i to q_1

Grasp Space Exploration

\mathbf{e}	tabletop plane Cartesian equation corresponding to the object stable pose
\mathbf{e}_r	tabletop plane Cartesian equation corresponding to the object release stable pose (in the variant proposed in Conclusions & Perspectives)
\mathbf{g}	gripper configuration

\mathbf{o} representation of the object geometry information

Q_{MSV} minimum singular value grasp quality metric

Variational Auto-Encoder

$\mathbf{X} \in \mathbb{R}^{n \times m}$ data matrix, with n samples of the random variable \mathbf{x} having each m features

$P(\mathbf{x}|\mathbf{z})$ conditional probability distribution of the observed random variable \mathbf{x} given a sample of the random latent variable \mathbf{z}

$P(\mathbf{x}|\mathbf{z}, \boldsymbol{\theta})$ the decoder part of a VAE: approximation of the unknown distribution $P(\mathbf{x}|\mathbf{z})$

$P(\mathbf{x})$ probability distribution of the observed random variable \mathbf{x}

$P(\mathbf{z})$ probability distribution of the latent random variable \mathbf{z}

$Q(\mathbf{z}|\mathbf{x}, \boldsymbol{\psi})$ the encoder part of a VAE: approximation of the unknown distribution $P(\mathbf{z}|\mathbf{x}, \boldsymbol{\theta})$

Acronyms

Brep Boundary representation.

CNC Computer Numerical Control.

CNN Convolutional Neural Network.

CSG Constructive Solid Geometry.

CVAE Conditional Variational Auto-Encoder.

GAN Generative Adversarial Network.

HGG Human-initiated Grasp Generator.

LLE Locally Linear Embedding.

MDS Multidimensional Scaling.

NN Neural Network.

PCA Principal Component Analysis.

QGG Quality-oriented Grasp Generator.

VAE Variational Auto-Encoder.

List of Figures

1	Examples of task specific industrial grippers, that use different physical principles to produce the grasp.	14
2	A cobot collaborating with an operator for a small parts assembly task [16]. Picture: copyright ©2021, ABB.	15
3	Organisation chart of the manuscript.	17
1.1	Various intralogistic operations that require grasping abilities.	21
1.2	ROMANS European project [30]. It aims at developing robotic solution for nuclear waste sorting and segregation. One of the proposed solution is a bi-manual master-slave tele-operated system.	22
1.3	Examples of vision sensors that can be used for grasping applications. . .	23
1.4	Example of a tool changer. Picture: copyright ©2021, DESTACO	23
1.5	Examples of suction grippers. For a suction element, the grasping force is given by $F = (P_a - P_v)A$, with A the contact surface, and $P_a - P_v$ the depression. P_a is the atmospheric pressure, and P_v the working pressure (a negative relative pressure).	25
1.6	Schemes of permanent magnet gripper principles.	26
1.7	Example of the two main grasp categories described in Napier et al. [36]. Pictures: copyright ©2016, Springer-Verlag Berlin Heidelberg [37]	30
1.8	Partial taxonomy of grasps encountered in a manufacturing context [39]. copyright ©1989, IEEE	31
1.9	Examples of dexterous and versatile multifingered robotic grippers in the literature.	33
1.10	Summary of the main tendon-based actuation schemes. The forces and torques transmitted from the actuator(s) (or spring in case of the passive return motion) to the joint in both directions are represented in green and orange respectively.	34
1.11	Scheme summarizing a human-cognition inspired multifingered gripper controller proposed in [38, 45].	35
1.12	Different types of underactuation.	37
1.13	Scheme of grasp planning approaches and their relative complexity and versatility.	41
1.14	General principle of grasp planners. The main focus and contribution of this thesis is boxed in red.	43

1.15	Scheme of an hypothetical two dimensional gripper configuration space, with the grasp space of a given object at a given position in the workspace.	47
1.16	Scheme of the contact point and gripper configuration approaches for grasp space exploration.	48
1.17	Grasping sequence of an underactuated finger using differential mechanisms [7]. Picture: copyright ©2008, Springer-Verlag Berlin Heidelberg	49
2.1	Geometric and kinematic description of a planar robotic finger, with all joints having the same rotation axis. In this particular case, $n_p = n_q$.	57
2.2	Effort applied by a finger during a grasp in the planar case.	58
2.3	Kinematic scheme of a robotic gripper with three two-phalanx fingers.	60
2.4	Scheme of a finger performing a precision grasp in the planar case.	61
2.5	Scheme of an underactuated finger performing a precision grasp in the planar case.	64
2.6	Scheme of three contacts on an object, produced by a three fingered gripper. O is the frame associated with the object, for example aligned on its center of mass and principle inertial axes. C_i is the frame associated with the i^{th} contact point: its origin on the contact point, and with the z axis along the contact normal and toward the object.	68
2.7	Diagram of the existing relations between quantities linked to the fingers, the contacts and the object.	70
2.8	A sphere grasped by a fully-actuated two fingered gripper with six joints.	75
2.9	A sphere grasped by an underactuated two fingered gripper with six joints. the underactuation mechanism acts on the two last joints of each fingers.	78
2.10	Example of an underactuated gripper with two two-phalanx fingers grasping a sphere. On the left a grasp configuration at equilibrium with a small tightening effort, on the right the grasp configuration at equilibrium after increasing the tightening force. The arrows show the contact normals, the dashed lines the orientation of the internal object forces. Tightening the grasp changes the orientation of the friction cone relative to the internal object forces.	81
3.1	Scheme of the considered setup and its associated frames.	90
3.2	Scheme of the considered underactuated gripper. This gripper has n_f underactuated fingers. Each finger has n_p underactuated phalanges, and k_i fully-actuated degrees of freedom allowing finger repositionning relative to the palm (which can include abduction-adduction motion or any other degrees of freedom).	91
3.3	The three stable poses of a right cuboid. For each stable pose, the two rows are two geometrically equivalent poses, that are due to symmetries in the object geometry. The orientation of the top row is chosen to express the Cartesian equation representing each stable pose.	94

3.4	Examples of successful grasps on an arbitrary 2D object, in the simple case of a bi-digital gripper with parallel jaws. Here, $d_{conf} = 3$, and the gripper configuration space is parameterized by three variables: (x, y, θ) .	94
3.5	Structure of an autoencoder.	99
3.6	Scheme of a feedforward neural network with one hidden layer.	101
3.7	Graphs of the three main activation functions.	101
3.8	Scheme of a Generative Adversarial Network architecture. In green the inputs, in blue the outputs and in red the terms of the loss function used to compute the gradient descent.	105
3.9	Directed graphical model of a VAE [135]. Solid lines describe the generative model $P(\mathbf{x} \mathbf{z}, \boldsymbol{\theta})P(\mathbf{z})$, and dashed lines represent the approximation $Q(\mathbf{z} \mathbf{x}, \boldsymbol{\phi})$ of the intractable distribution $P(\mathbf{z} \mathbf{x}, \boldsymbol{\theta})$. The parameter $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$ are learned simultaneously by sampling in the dataset \mathbf{X} . One can sample n times a \mathbf{x} or \mathbf{z} with fixed $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$.	110
3.10	Scheme of the training process of the VAE, with the reparameterization trick. In green the encoder and decoder inputs, in blue their outputs, and in red the terms of the loss function used for the gradient descent.	111
3.11	Scheme of a VAE after the training, when used to generate new data: only the decoder part is used, and the latent variables are sampled from $P(\mathbf{z}) = \mathcal{N}(\mathbf{0}, \mathbf{I})$. In green the decoder input, and in blue its output.	112
3.12	Scheme of the training process of the CVAE, with the reparameterization trick. In green the encoder and decoder inputs, in blue their outputs, and in red the terms of the loss function used for the gradient descent.	113
3.13	Scheme of a CVAE after the training, when used to generate new data: only the decoder part is used, and the latent variables are sampled from $P(\mathbf{z}) = \mathcal{N}(\mathbf{0}, \mathbf{I})$. In green the decoder input, and in blue its output.	114
3.14	Scheme of the presented workflow.	115
3.15	Example of two grasp configurations belonging to two different grasp types on a cylinder, for a two-fingered gripper with two phalanges.	117
3.16	Scheme of the training workflow of the HGG, with \mathbf{g} the grasp configuration, and \mathbf{e} the tabletop Cartesian equation as defined in subsection 3.1.2. In green the inputs, in blue the outputs and in red the loss functions used during the training.	118
3.17	Scheme of the training workflow of the QGG, with \mathbf{g} the grasp configuration and \mathbf{e} the tabletop Cartesian equation as defined in subsection 3.1.2, and Q_{MSV} the grasp quality. In green the inputs, in blue the outputs, and in red the loss functions used during the training.	119
4.1	Picture and actuation scheme of the CEA three-fingered underactuated gripper used in this work.	122
4.2	Gripper and camera mounted on an ABB IRB 4600 industrial robot arm.	123
4.3	Gripper frame F_{grip} relative to the object frame F_{obj} , and spread angle θ .	124

4.4	Example of a grasp with four contacts: three contacts with the fingertips c_1, c_2, c_3 , and one contact with the palm, not represented as hidden by the object. FC_i is the friction cone of the contact i , and f_{c-i} the contact force for the contact i	124
4.5	The chosen objects and their frame F_{obj} in their different stable poses. . .	125
4.6	Primitive grasp types for the three chosen objects.	126
4.7	HGG and QGG architecture. In blue the input layers, in green the hidden Neural Network (NN) and in red the output layers. The hidden NN inner layers are fully connected layers, with hyperbolic tangent activation functions. The main encoding and main decoding NN have symmetrical inner architecture. The inputs and outputs NN are small networks (with much less parameters compared to main encoding and decoding NN) in charge of extracting or reconstructing specific features associated with position, orientation, spread angle, tabletop equation or quality respectively. The supplementary input for the tabletop plane Cartesian equation ensures that the generated grasp depends on it [147]. The tabletop input NN appears both in encoder and decoder: it is the same network in both places (same weights), and not two different networks. This architecture is implemented with Tensorflow [156] and Keras [155] python libraries. The QGG has the same architecture and hyperparameters, with an added output to the decoder squared in orange.	128
4.8	Gripper configurations generated when visiting a two dimensional latent space of an HGG for the bent pipe. The image inside the red square is the point $(0, 0)$ in latent space. The images between the red and blue square and the images between the blue and green square correspond to points evenly distributed on circles of diameter respectively 0.5 and 1 in latent space. The coordinates of these configurations are shown with black cross markers on Figure 4.9. Here, translations along the image plan normal are not visible, which explains some visually almost identical configurations. . .	130
4.9	Colormap of the latent space of an HGG for the bent pipe object, for one stable pose, when using a two dimensional latent space (latent variables l_1 and l_2). The black cross markers correspond to coordinates of configurations displayed on Figure 4.8. The color value is the output of the HGG decoder for the corresponding l_1 and l_2 values. Values for x , y and z components are in meters (and without dimension for the other components).	131
4.10	Area for object positioning. For each trial, the object pose is chosen randomly inside the blue rectangle.	142
4.11	Pictures of eight grasps generated by the QGG and executed on the real setup after following the grasp planning procedure describer in Algorithm 1.143	
5.1	Examples of set Boolean operations used in CSG on a block and sphere primitive shapes. pictures: CC-BY-SA license [160]	146

5.2	Example of the graph of a mesh boundary representation of a cube, using square mesh faces. On the left, the mesh of the cube is depicted, with each vertex numbered. On the right, each vertex is represented as a blue node, with its corresponding spatial position between parentheses.	148
5.3	Scheme of the training workflow of the geom-QGG, with \mathbf{g} the grasp configuration, \mathbf{e} the tabletop Cartesian equation, and Q_{MSV} the grasp quality as defined in subsection 3.1.2, and \mathbf{o} a representation of the object geometry. In green the inputs, in blue the outputs, and in red the loss functions used during the training.	149
5.4	Scheme of the proposed workflow to take into account the object geometry in the grasp space exploration.	151
5.5	Adjacency, degree and Laplacian matrices of a graph, noted \mathbf{A} , \mathbf{D} and \mathbf{L} respectively.	152
5.6	Scheme of the one-ring triangles of a vertex \mathbf{v}_i of a mesh. V_{ii} is the area of the Voronoi area drawn in red.	153
5.7	Four object from the ShapeNetCore dataset, with their spectral clustering and corresponding approximation with ellipsoids and cuboids.	154
5.8	Example of deformations applied to the mesh to constitute the dataset: here an anisotropic scaling and a shearing is applied on both mesh.	155
5.9	Architecture of the geometry autoencoder trained on the dataset based on vertices sampling. In dark blue the input layer, in light blue the processing layers with no trainable weights, in green the hidden layers, and in red the output layer. d is the depth of the encoder and the decoder (that is the number of fully connected layers), and n is the number of features of the input vector.	157
5.10	Object chosen for the restricted dataset, to test if the failed learning is due to a too great complexity of the previous dataset based on ShapeNet.	158
5.11	Architecture of the geometry autoencoder trained on the dataset based on object geometry remeshing. In dark blue the input layer, in light blue the processing layers with no trainable weights, in green the hidden layers, and in red the output layer. d is the depth of the encoder and the decoder (that is the number of fully connected layers), l is the latent space dimension, and w is a width factor of the fully connected layers of the encoder and decoder.	160
5.12	Comparison between the reconstructed meshes (green) and input meshes (blue) for a model with $l = 32$, $d = 3$ and $w = 1$. The applied transformations on the meshes depicted here are different from the one presents in the training, validation and test sets.	161
5.13	Loss value obtained during the learning process of the geometry autoencoder, with a latent space dimension $l = 32$, a depth $d = 3$, and a width factor $w = 1$	162

5.14	Loss value obtained during the learning process of the geometry autoencoder, with a latent space dimension $l = 16$, a depth $d = 3$, and a width factor $w = 2$. Compared to the training graph shown in Figure 5.13, this training is obtained with an initial learning rate divided by a factor 2 relative to the default one, an exponential learning rate decay, and a dataset augmentation with reduced variance on translation and scaling.	163
5.15	Comparison between the reconstructed meshes (green) and input meshes (blue) for a model with $l = 16$, $d = 4$ and $w = 2$, and a KL divergence loss coefficient of 10^{-7} . The applied transformations on the meshes depicted here are different from the one presents in the training, validation and test sets.	164
5.16	Geom-QGG architecture. In blue the input layers, in green the hidden Neural Network (NN) and in red the output layers. The hidden NN inner layers are fully connected layers, with hyperbolic tangent or leaky relu activation functions. The main encoding and main decoding NN have symmetrical inner architecture. The inputs and outputs NN are small networks (with much less parameters compared to main encoding and decoding NN) in charge of extracting or reconstructing specific features associated with position, orientation, spread angle, tabletop equation or quality respectively. The supplementary input for the tabletop plane Cartesian equation and compressed geometry ensures that the generated grasp depends on them [147]. The tabletop input NN appears both in encoder and decoder: it is the same network in both places (same weights), and not two different networks. The compressed geometry is obtained through the latent space of the geometry encoder. This architecture is implemented with Tensorflow [156] and Keras [155] python libraries.	166
5.17	Examples of gripper configurations with their latent space coordinates, generated for non-learnt objects in latent space areas that are likely to produce successful grasps. these areas are found by manual inspection of the latent space.	168
5.18	Comparison between the original version of the cinder block, and a version modified by applying an isotropic scaling of factor 1.1.	169
4	Use of virtual reality to specify primitive grasps.	175
5	Use of a collaborative robot to specify primitive grasps. When in gravity compensation mode, the operator is able to move the robot arm end-effector in any desired configuration (here with a Kuka iiwa): it could be used to set the primitive grasps. Picture: copyright ©2022 KUKA.	175
6	Example in a two dimensional cut of the interpolation of a grasp type area from six primitive grasps, visible in blue. The blue crosses represents their positions, and the blue arrows the gripper palm orientation at these positions.	176

7	Case when the initial grasp stable pose is different from the release stable pose. Here, the grasp candidate 2 is not compatible with the release stable pose due to collision with the tabletop surface.	178
8	Possible implementation of a QGG taking into account the object release pose. In green the inputs, in blue the outputs, and in red the loss function components. \mathbf{e} is the initial grasp stable pose, and \mathbf{e}_r is the additional condition corresponding to the release pose.	178
1	Scheme of a finger of the considered underactuated gripper in chapter 4. .	182
2	Scheme of the global control architecture used on the real robotic setup. .	183

List of Tables

1.1	Comparison of the ability of grippers to pick up different types of objects (●●: commonly used, ●: sometime used) [35].	28
2.1	Summary of common contact types (in the case of a 3D contact). The contact wrench is expressed in the contact frame, its z axis being toward the contact normal.	56
2.2	Summary of the contact wrench and twist vectors in the 2D and 3D cases.	59
2.3	Summary of the Jacobian matrices in the 2D and 3D cases.	63
2.4	Summary of main grasp categories [37].	72
2.5	Summary of the main existing categories of grasp quality metrics.	86
4.1	Smallest number of dimensions allowing to achieve a reconstruction error at most twice the one obtained with seven dimensions.	134
4.2	Spearman correlation coefficients between the hyperparameters and the indicators.	135
4.3	performances of the HGG for the selected hyperparameters: 30 000 network parameters, 3 latent variables (that are all used), and a KL divergence coefficient of 0.0005. The mean errors are measured on the training data.	136
4.4	Information summary about primitive and HGG generated grasps (step 3 of the workflow in Figure 3.14) for the selected hyperparameters: 30 000 network parameters, 3 latent variables (that are all used), and a KL divergence coefficient of 0.0005. The metric statistics are taking into account successful grasps only.	136
4.5	Comparison of main characteristics of grasp space exploration methods.	138
4.6	performances summary of a sampling-based grasp space exploration method. The mean and median quality metric values are computed on the successful grasps only.	140
4.7	Performances on simulated grasp planning trials.	141
5.1	Performances of the geom-QGG on simulated grasp planning trials.	167

5.2 Comparison between grasps generated for the original and deformed cinder blocks. First, grasps are selected in the original cinder block latent space (left). Then, the goal is to find grasps resembling to them in the deformed cinder block latent space (right). In the middle is displayed the grasps generated for the deformed cinder block at the same latent space coordinates as the selected grasp on the left. 171

Bibliography

- [1] M. Hägele, K. Nilsson, J. N. Pires, and R. Bischoff. “Industrial Robotics”. In: *Springer Handbook of Robotics*. Ed. by B. Siciliano and O. Khatib. Springer Handbooks. Cham: Springer International Publishing, 2016, pp. 1385–1422. ISBN: 978-3-319-32552-1. DOI: 10.1007/978-3-319-32552-1_54 (cit. on pp. 7, 13, 40, 42).
- [2] J.-P. Gazeau. *Préhension robotique et manipulation dextre*. Techniques de l’Ingénieur, 2020 (cit. on pp. 7, 9, 13, 16, 24, 25, 26, 27, 29, 30, 32, 34).
- [3] European Commission. Directorate General for Research and Innovation. *Industry 5.0: Towards a Sustainable, Human Centric and Resilient European Industry*. Publications Office, 2021 (cit. on pp. 8, 14).
- [4] S. Nahavandi. “Industry 5.0—A Human-Centric Solution”. In: *Sustainability* 11.16 (16 Jan. 2019), p. 4371. DOI: 10.3390/su11164371 (cit. on pp. 8, 14, 29).
- [5] S. Bragança, E. Costa, I. Castellucci, and P. M. Arezes. “A Brief Overview of the Use of Collaborative Robots in Industry 4.0: Human Role and Safety”. In: *Occupational and Environmental Safety and Health*. Ed. by P. M. Arezes et al. Vol. 202. Studies in Systems, Decision and Control. Cham: Springer International Publishing, 2019, pp. 641–650. DOI: 10.1007/978-3-030-14730-3_68 (cit. on pp. 8, 14).
- [6] W. Townsend. “The BarrettHand grasper – programmably flexible part handling and assembly”. In: *Industrial Robot* 27.3 (2000), pp. 181–188. ISSN: 0143-991X. DOI: 10.1108/01439910010371597 (cit. on pp. 8, 15, 29, 37, 38, 49).
- [7] L. Birglen, T. Laliberté, and C. M. Gosselin. *Underactuated Robotic Hands*. Ed. by B. Siciliano, O. Khatib, and F. Groen. Springer Tracts in Advanced Robotics 40. Springer, Dec. 23, 2007. 248 pp. ISBN: 978-3-540-77459-4 (cit. on pp. 9, 16, 32, 34, 36, 37, 38, 49, 53, 61, 62, 64, 65, 66, 78).
- [8] C. Rolinat, M. Grossard, S. Aloui, and C. Godin. “Human Initiated Grasp Space Exploration Algorithm for an Underactuated Robot Gripper Using Variational Autoencoder”. In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. 2021, pp. 2598–2604. DOI: 10.1109/ICRA48506.2021.9561765 (cit. on pp. 11, 18).

- [9] C. Rolinat, M. Grossard, S. Aloui, and C. Godin. “Learning to Model the Grasp Space of an Underactuated Robot Gripper Using Variational Autoencoder”. In: *IFAC-PapersOnLine*. 19th IFAC Symposium on System Identification SYSID 2021 54.7 (2021), pp. 523–528. ISSN: 2405-8963. DOI: 10.1016/j.ifacol.2021.08.413 (cit. on pp. 11, 18).
- [10] C. Rolinat, M. Grossard, S. Aloui, and C. Godin. “Grasp Space Exploration Method for an Underactuated Gripper Using Human Initiated Primitive Grasps”. Submitted to International Journal of Intelligent Robotics and Applications, under review (cit. on pp. 11, 18).
- [11] C. Rolinat, M. Grossard, S. Aloui, and C. Godin. “Méthode de génération de données pour la commande d’un préhenseur d’un système robotisé”. Pat. FR2011310. CEA. Nov. 4, 2020 (cit. on pp. 11, 18).
- [12] KUKA. *KR 500-3 handles railway sleepers at Maba Track Solutions GmbH*. July 2016. URL: <https://www.kuka.com/en-gb/industries/solutions-database/2016/07/solution-robotics-maba> (cit. on p. 14).
- [13] Goudsmit. *Ventouse pneumo-magnétique pour aimant de manipulation*. Apr. 2020. URL: <https://www.goudsmitmagnets.com/fr/annoncements/news/magnetic-gripper-for-robot-end-of-arm-tooling> (cit. on p. 14).
- [14] KUKA. *Assembly line work 4.0 – thanks to KUKA palletizing robot*. Nov. 2020. URL: <https://www.kuka.com/en-gb/industries/solutions-database/2020/10/scott> (cit. on p. 14).
- [15] ABB. *Robot-powered intelligent selection for automotive batteries boosts daily output by 300 percent*. July 2021. URL: <https://new.abb.com/news/detail/79200/cstmr-robot-powered-intelligent-selection-for-automotive-batteries-boosts-daily-output-by-300-percent> (cit. on p. 14).
- [16] ABB. *YuMi manufacturing sockets at ABB’s plant in the Czech Republic*. Apr. 2017. URL: <https://new.abb.com/news/detail/62029/yumi-manufacturing-sockets-at-abbs-plant-in-the-czech-republic> (cit. on p. 15).
- [17] R. Bloss. “Review of Manufacturing Cells as They Achieve High Levels of Autonomy and Flexibility”. In: *Assembly Automation* 33.2 (Jan. 1, 2013), pp. 112–116. ISSN: 0144-5154. DOI: 10.1108/01445151311306618 (cit. on p. 20).
- [18] CollaborativePalletizer. *Collaborative-Palletizer-AAA20-RAAS-Universal Robot*. URL: https://commons.wikimedia.org/wiki/File:Collaborative-Palletizer-AAA20-RAAS-Universal_Robot.jpg (cit. on p. 21).
- [19] RECFI Industrie. *BIN PACKING SYSTEMS*. URL: <https://emballage-manutention.recfi-industrie.com/en/packing-solutions/74-bin-packing-systems.html> (cit. on p. 21).
- [20] ROMI Industrial Systems. *Robotics*. URL: www.romi-is.com/?page_id=42 (cit. on p. 21).

-
- [21] EXOTEC. *Elegant warehouse robotics for a volatile world*. URL: www.exotec.com/en/home/ (cit. on p. 21).
- [22] LAC Conveyors & Automation. *The Skypicker*. URL: www.lacconveyors.co.uk/products/exotec-skypicker/ (cit. on p. 21).
- [23] A. P. Kahn. *The Encyclopedia of Work-Related Illnesses, Injuries, and Health Issues*. Infobase Publishing, 2004. 449 pp. ISBN: 978-0-8160-6628-5 (cit. on p. 21).
- [24] D. Rossi, E. Bertoloni, M. Fenaroli, F. Marciano, and M. Alberti. “A Multi-Criteria Ergonomic and Performance Methodology for Evaluating Alternatives in “Manuable” Material Handling”. In: *International Journal of Industrial Ergonomics* 43.4 (July 1, 2013), pp. 314–327. ISSN: 0169-8141. DOI: 10.1016/j.ergon.2013.04.009 (cit. on p. 21).
- [25] S. Charbonnier, R. N. Roy, S. Bonnet, and A. Campagne. “EEG index for control operators’ mental fatigue monitoring using interactions between brain regions”. In: *Expert Systems with Applications* 52 (June 15, 2016), pp. 91–98. ISSN: 0957-4174. DOI: 10.1016/j.eswa.2016.01.013 (cit. on p. 21).
- [26] M. Wilson. “Developments in Robot Applications for Food Manufacturing”. In: *Industrial Robot: An International Journal* 37.6 (Jan. 1, 2010), pp. 498–502. ISSN: 0143-991X. DOI: 10.1108/01439911011081632 (cit. on p. 22).
- [27] M. R. J. Moreno, J. O. Gray, T. J. Dodd, and D. G. Caldwell. “Guidelines for the Design of Low-cost Robots for the Food Industry”. In: *Industrial Robot: An International Journal* 37.6 (Jan. 1, 2010), pp. 509–517. ISSN: 0143-991X. DOI: 10.1108/01439911011081650 (cit. on p. 22).
- [28] K. Mathia. *Robotics for Electronics Manufacturing: Principles and Applications in Cleanroom Automation*. Cambridge University Press, May 6, 2010. 251 pp. ISBN: 978-0-521-87652-0. Google Books: TKFrZ2rpkU0C (cit. on p. 22).
- [29] R. Bogue. “Robots in the Nuclear Industry: A Review of Technologies and Applications”. In: *Industrial Robot: An International Journal* 38.2 (Jan. 1, 2011), pp. 113–118. ISSN: 0143-991X. DOI: 10.1108/01439911111106327 (cit. on p. 22).
- [30] R. Stolkin et al. *Robotic Manipulation for Nuclear Sort and Segregation*. 2015. URL: www.h2020romans.org (cit. on p. 22).
- [31] G. Du, K. Wang, S. Lian, and K. Zhao. “Vision-Based Robotic Grasping from Object Localization, Object Pose Estimation to Grasp Estimation for Parallel Grippers: A Review”. In: *Artificial Intelligence Review* 54.3 (Mar. 1, 2021), pp. 1677–1734. ISSN: 1573-7462. DOI: 10.1007/s10462-020-09888-5 (cit. on pp. 23, 40, 41, 42, 92).
- [32] K. Kleeberger, R. Bormann, W. Kraus, and M. F. Huber. “A Survey on Learning-Based Robotic Grasping”. In: *Current Robotics Reports* 1.4 (2020), pp. 239–249. DOI: 10.1007/s43154-020-00021-6 (cit. on pp. 23, 40).
- [33] G. J. Monkman, S. Hesse, R. Steinmann, and H. Schunk. *Robot Grippers*. John Wiley & Sons, Feb. 27, 2007. 466 pp. ISBN: 978-3-527-60989-5 (cit. on p. 24).

- [34] G.J. Monkman. “An Analysis of Astrictive Prehension”. In: *The International Journal of Robotics Research* 16.1 (Feb. 1, 1997). Publisher: SAGE Publications Ltd STM, pp. 1–10. ISSN: 0278-3649. DOI: 10.1177/027836499701600101 (cit. on pp. 25, 26).
- [35] K. Tai, A.-R. El-Sayed, M. Shahriari, M. Biglarbegian, and S. Mahmud. “State of the Art Robotic Grippers and Applications”. In: *Robotics* 5.2 (June 2016). Number: 2 Publisher: Multidisciplinary Digital Publishing Institute, p. 11. DOI: 10.3390/robotics5020011 (cit. on p. 28).
- [36] J. Napier, J.R. Napier, R. Tuttle, and R.H. Tuttle. *Hands*. Natural Science. Princeton University Press, 1993. ISBN: 978-0-691-02547-6 (cit. on p. 30).
- [37] D. Prattichizzo and J. C. Trinkle. “Grasping”. In: *Springer Handbook of Robotics*. Ed. by B. Siciliano and O. Khatib. Springer Handbooks. Cham: Springer International Publishing, 2016, pp. 955–988. ISBN: 978-3-319-32552-1. DOI: 10.1007/978-3-319-32552-1_38 (cit. on pp. 30, 53, 54, 61, 62, 63, 68, 69, 71, 72, 73, 84).
- [38] M. Monforte, F. Ficuciello, and B. Siciliano. “Human Cognition-Inspired Robotic Grasping”. In: *Cognitive Architectures*. Ed. by M. I. Aldinhas Ferreira, J. Silva Sequeira, and R. Ventura. Vol. 94. Cham: Springer International Publishing, 2019, pp. 71–84. DOI: 10.1007/978-3-319-97550-4_6 (cit. on pp. 30, 32, 35).
- [39] M.R. Cutkosky. “On Grasp Choice, Grasp Models, and the Design of Hands for Manufacturing Tasks”. In: *IEEE Transactions on Robotics and Automation* 5.3 (June 1989), pp. 269–279. ISSN: 2374-958X. DOI: 10.1109/70.34763 (cit. on p. 31).
- [40] I. M. Bullock and A. M. Dollar. “Classifying human manipulation behavior”. In: *2011 IEEE International Conference on Rehabilitation Robotics*. ISSN: 1945-7901. June 2011, pp. 1–6. DOI: 10.1109/ICORR.2011.5975408 (cit. on p. 30).
- [41] T. Feix, J. Romero, H.-B. Schmiebmayer, A. M. Dollar, and D. Kragic. “The GRASP Taxonomy of Human Grasp Types”. In: *IEEE Transactions on Human-Machine Systems* 46.1 (Feb. 2016), pp. 66–77. ISSN: 2168-2305. DOI: 10.1109/THMS.2015.2470657 (cit. on p. 30).
- [42] J. Romero. “From Human to Robot Grasping”. PhD thesis. KTH Computer Science and Communication, 2011 (cit. on pp. 32, 35).
- [43] A. Bicchi. “Hands for Dexterous Manipulation and Powerful Grasping: A Difficult Road Towards Simplicity”. In: *Robotics Research*. Ed. by G. Giralt and G. Hirzinger. London: Springer London, 1996, pp. 2–15. ISBN: 978-1-4471-1021-7. DOI: 10.1007/978-1-4471-1021-7_2 (cit. on p. 32).
- [44] Shadow Robot Company. *Shadow Hand*. URL: https://commons.wikimedia.org/wiki/File:Shadow_Hand.jpg (visited on Feb. 21, 2022) (cit. on p. 33).

- [45] G. Palli et al. “The DEXMART hand: Mechatronic design and experimental evaluation of synergy-based control for human-like grasping”. In: *The International Journal of Robotics Research* 33.5 (Apr. 1, 2014). Publisher: SAGE Publications Ltd STM, pp. 799–824. ISSN: 0278-3649. DOI: 10.1177/0278364913519897 (cit. on pp. 33, 35).
- [46] H. Liu et al. “The modular multisensory DLR-HIT-Hand”. In: *Mechanism and Machine Theory* 42.5 (May 2007), pp. 612–625. ISSN: 0094114X. DOI: 10.1016/j.mechmachtheory.2006.04.013 (cit. on p. 33).
- [47] P. Vulliez, J. P. Gazeau, P. Laguillaumie, H. Mnyusiwalla, and P. Seguin. “Focus on the Mechatronics Design of a New Dexterous Robotic Hand for inside Hand Manipulation”. In: *Robotica* 36.8 (Aug. 2018), pp. 1206–1224. ISSN: 0263-5747, 1469-8668. DOI: 10.1017/S0263574718000346 (cit. on pp. 33, 34).
- [48] Mathieu Grossard, Nicolas Chaillet, and Stéphane Régnier, eds. *Flexible robotics: applications to multiscale manipulations*. OCLC: ocn814372743. London, UK : Hoboken, NJ: ISTE Ltd ; John Wiley & Sons, Inc, 2013. 384 pp. ISBN: 978-1-84821-520-7 (cit. on pp. 32, 34).
- [49] M. T. Mason and J. K. Salisbury. *Robot Hands and the Mechanics of Manipulation*. In collab. with MIT Press. Cambridge, Mass. : MIT Press, 1985. 338 pp. ISBN: 978-0-262-13205-3 (cit. on p. 34).
- [50] G. Gioioso, G. Salvietti, M. Malvezzi, and D. Prattichizzo. “Mapping Synergies From Human to Robotic Hands With Dissimilar Kinematics: An Approach in the Object Domain”. In: *IEEE Transactions on Robotics* 29.4 (Aug. 2013), pp. 825–837. ISSN: 1552-3098, 1941-0468. DOI: 10.1109/TR0.2013.2252251 (cit. on p. 35).
- [51] T. Laliberté and C. Gosselin. “Actuation System for Highly Underactuated Gripping Mechanism”. U.S. pat. 6505870B1. Université Laval. Jan. 14, 2003 (cit. on pp. 37, 38).
- [52] *Laboratoire de robotique: Main SARAH (préhenseur sous-actionné pour l'espace)*. URL: <https://robot.gmc.ulaval.ca/recherche/theme-de-recherche/mains-et-prehenseurs/main-sarah-prehenseur-sous-actionne-pour-lespace/> (visited on Feb. 19, 2022) (cit. on p. 37).
- [53] R. Deimel and O. Brock. “A Novel Type of Compliant and Underactuated Robotic Hand for Dexterous Grasping”. In: *The International Journal of Robotics Research* 35.1-3 (Aug. 21, 2015), pp. 161–185. DOI: 10.1177/0278364915592961 (cit. on pp. 37, 38).
- [54] *Barrett Hand*. robots.ros.org. URL: <https://robots.ros.org/barrett-hand/> (visited on Feb. 19, 2022) (cit. on p. 37).
- [55] S. Krut. “A Force-Isotropic Underactuated Finger”. In: *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*. Apr. 2005, pp. 2314–2319. DOI: 10.1109/ROBOT.2005.1570458 (cit. on p. 37).

- [56] A Sahbani, S El-Khoury, and P Bidaud. “An Overview of 3D Object Grasp Synthesis Algorithms”. In: *Robotics and Autonomous Systems* 60.3 (2012), pp. 326–336 (cit. on pp. 40, 42).
- [57] M. A. c. Riedlinger, M. Voelk, K. Kleeberger, M. U. Khalid, and R. Bormann. “Model-Free Grasp Learning Framework based on Physical Simulation”. In: *ISR 2020; 52th International Symposium on Robotics*. 2020, pp. 1–8 (cit. on pp. 40, 44, 45, 48, 49, 138, 139).
- [58] S. Salti, F. Tombari, and L. Di Stefano. “SHOT: Unique Signatures of Histograms for Surface and Texture Description”. In: *Computer Vision and Image Understanding* 125 (Aug. 1, 2014), pp. 251–264. ISSN: 1077-3142. DOI: 10.1016/j.cviu.2014.04.011 (cit. on pp. 42, 44, 92).
- [59] J. Yang, H. Li, D. Campbell, and Y. Jia. “Go-ICP: A Globally Optimal Solution to 3D ICP Point-Set Registration”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 38.11 (Nov. 2016), pp. 2241–2254. ISSN: 1939-3539. DOI: 10.1109/TPAMI.2015.2513405 (cit. on pp. 42, 44, 92).
- [60] B. Drost, M. Ulrich, N. Navab, and S. Ilic. “Model globally, match locally: Efficient and robust 3D object recognition”. In: *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. 2010, pp. 998–1005 (cit. on pp. 42, 44, 92).
- [61] D. Buchholz, M. Futterlieb, S. Winkelbach, and F. M. Wahl. “Efficient Bin-Picking and Grasp Planning Based on Depth Data”. In: *2013 IEEE International Conference on Robotics and Automation*. May 2013, pp. 3245–3250. DOI: 10.1109/ICRA.2013.6631029 (cit. on p. 42).
- [62] D. Berenson, R. Diankov, K. Nishiwaki, S. Kagami, and J. Kuffner. “Grasp planning in complex scenes”. In: *IEEE-RAS International Conference on Humanoid Robots*. 2007, pp. 42–48 (cit. on pp. 42, 44, 45).
- [63] M. A. Roa, R. Suarez, and J. Rosell. “Grasp space generation using sampling and computation of independent regions”. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2008, pp. 2258–2263 (cit. on pp. 42, 44, 45, 47, 49).
- [64] Z. Xue, J. M. Zoellner, and R. Dillmann. “Grasp planning: Find the contact points”. In: *IEEE International Conference on Robotics and Biomimetics (RO-BIO)*. 2007, pp. 835–840 (cit. on pp. 42, 44, 45, 47).
- [65] S. El-Khoury and A. Sahbani. “On computing robust n-finger force-closure grasps of 3D objects”. In: *IEEE International Conference on Robotics and Automation*. 2009, pp. 2480–2486. DOI: 10.1109/ROBOT.2009.5152272 (cit. on pp. 42, 44, 45, 47, 49).

-
- [66] J. Ponce, S. Sullivan, J.-D. Boissonnat, and J.-P. Merlet. “On Characterizing and Computing Three- and Four-Finger Force-Closure Grasps of Polyhedral Objects”. In: *[1993] Proceedings IEEE International Conference on Robotics and Automation*. Vol. 2. May 1993, pp. 821–827. DOI: 10.1109/ROBOT.1993.291933 (cit. on pp. 42, 44, 45, 47, 49).
- [67] G. Recatalá, E. Chinellato, Á. P. del Pobil, Y. Mezouar, and P. Martinet. “Biologically-Inspired 3D Grasp Synthesis Based on Visual Exploration”. In: *Autonomous Robots* 25.1 (Aug. 1, 2008), pp. 59–70. ISSN: 1573-7527. DOI: 10.1007/s10514-008-9086-7 (cit. on pp. 42, 44).
- [68] L. Pinto and A. Gupta. “Supersizing Self-supervision: Learning to Grasp from 50K Tries and 700 Robot Hours”. In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2016, pp. 3406–3413. arXiv: 1509.06825 (cit. on pp. 44, 45, 48, 49, 138).
- [69] A. Depierre, E. Dellandréa, and L. Chen. “Jacquard: A Large Scale Dataset for Robotic Grasp Detection”. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Oct. 2018, pp. 3511–3516. DOI: 10.1109/IROS.2018.8593950 (cit. on pp. 44, 45, 48, 49, 138).
- [70] S. Levine, P. Pastor, A. Krizhevsky, J. Ibarz, and D. Quillen. “Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection”. In: *The International Journal of Robotics Research* 37.4 (2018), pp. 421–436. ISSN: 0278-3649, 1741-3176. DOI: 10.1177/0278364917710318 (cit. on pp. 44, 45, 48, 49, 138).
- [71] A. Mousavian, C. Eppner, and D. Fox. “6-DOF GraspNet: Variational Grasp Generation for Object Manipulation”. In: *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*. 2019, pp. 2901–2910. DOI: 10.1109/ICCV.2019.00299 (cit. on pp. 44, 45, 48, 49, 138, 139).
- [72] Z. Zhao, W. Shang, H. He, and Z. Li. “Grasp Prediction and Evaluation of Multi-Fingered Dexterous Hands Using Deep Learning”. In: *Robotics and Autonomous Systems* 129 (July 2020), p. 103550. ISSN: 09218890. DOI: 10.1016/j.robot.2020.103550 (cit. on pp. 44, 47, 49).
- [73] R. Pelossof, A. Miller, P. Allen, and T. Jebara. “An SVM learning approach to robotic grasping”. In: *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA. Vol. 4*. New Orleans, LA, USA: IEEE, Apr. 2004, pp. 3512–3518. ISBN: 978-0-7803-8232-9. DOI: 10.1109/ROBOT.2004.1308797 (cit. on pp. 44, 50).
- [74] J. Mahler et al. “Dex-Net 2.0: Deep Learning to Plan Robust Grasps with Synthetic Point Clouds and Analytic Grasp Metrics”. In: *Robotics: Science and Systems (RSS)*. 2017. arXiv: 1703.09312 (cit. on pp. 44, 45, 48, 49).

- [75] C. Choi, W. Schwarting, J. DelPreto, and D. Rus. “Learning Object Grasping for Soft Robot Hands”. In: *IEEE Robotics and Automation Letters* 3.3 (July 2018), pp. 2370–2377. ISSN: 2377-3766, 2377-3774. DOI: 10.1109/LRA.2018.2810544 (cit. on p. 50).
- [76] C. D. Santina et al. “Learning From Humans How to Grasp: A Data-Driven Architecture for Autonomous Grasping With Anthropomorphic Soft Hands”. In: *IEEE Robotics and Automation Letters* 4.2 (2019), pp. 1533–1540. ISSN: 2377-3766. DOI: 10.1109/LRA.2019.2896485 (cit. on p. 50).
- [77] R. M. Murray, Z. Li, and S. S. Sastry. *A Mathematical Introduction to Robotic Manipulation*. CRC Press, 1994 (cit. on pp. 53, 54, 61, 62, 63, 68, 69, 73).
- [78] D. Prattichizzo and A. Bicchi. “Dynamic Analysis of Mobility and Graspability of General Manipulation Systems”. In: *IEEE Transactions on Robotics and Automation* 14.2 (Apr. 1998), pp. 241–258. ISSN: 2374-958X. DOI: 10.1109/70.681243 (cit. on p. 53).
- [79] A. Bicchi. “On the Closure Properties of Robotic Grasping”. In: *The International Journal of Robotics Research* 14.4 (Aug. 1, 1995), pp. 319–334. ISSN: 0278-3649. DOI: 10.1177/027836499501400402 (cit. on p. 53).
- [80] L. Birglen and C. Gosselin. “On the Force Capability of Underactuated Fingers”. In: *2003 IEEE International Conference on Robotics and Automation*. Vol. 1. Sept. 2003, pp. 1139–1145. DOI: 10.1109/ROBOT.2003.1241746 (cit. on p. 53).
- [81] L. Birglen and C. Gosselin. “Kinetostatic Analysis of Underactuated Fingers”. In: *IEEE Transactions on Robotics and Automation* 20.2 (Apr. 2004), pp. 211–221. ISSN: 2374-958X. DOI: 10.1109/TRA.2004.824641 (cit. on p. 53).
- [82] L. Birglen and C. Gosselin. “Optimal Design of 2-Phalanx Underactuated Fingers”. In: *Proceedings of 2004 International Conference on Intelligent Manipulation and Grasping*. July 2004, pp. 110–116 (cit. on p. 53).
- [83] D. Prattichizzo, M. Malvezzi, and A. Bicchi. “On motion and force controllability of grasping hands with postural synergies”. In: *Robotics: Science and Systems*. Vol. 6. 2010, pp. 49–56 (cit. on p. 53).
- [84] D. Prattichizzo, M. Malvezzi, M. Gabiccini, and A. Bicchi. “On the Manipulability Ellipsoids of Underactuated Robotic Hands with Compliance”. In: *Robotics and Autonomous Systems* 60.3 (Mar. 2012), pp. 337–346. ISSN: 09218890. DOI: 10.1016/j.robot.2011.07.014 (cit. on p. 53).
- [85] I. Kao, K. M. Lynch, and J. W. Burdick. “Contact Modeling and Manipulation”. In: *Springer Handbook of Robotics*. Ed. by B. Siciliano and O. Khatib. Springer Handbooks. Cham: Springer International Publishing, 2016, pp. 931–954. ISBN: 978-3-319-32552-1. DOI: 10.1007/978-3-319-32552-1_37 (cit. on p. 54).
- [86] M. A. Roa and R. Suárez. “Grasp quality measures: review and performance”. In: *Autonomous Robots* 38.1 (2015), pp. 65–88. ISSN: 0929-5593, 1573-7527. DOI: 10.1007/s10514-014-9402-3 (cit. on pp. 71, 82, 83, 85, 86, 87).

-
- [87] Z. Li and S.S. Sastry. “Task-Oriented Optimal Grasping by Multifingered Robot Hands”. In: *IEEE Journal on Robotics and Automation* 4.1 (Feb. 1988), pp. 32–44. ISSN: 2374-8710. DOI: 10.1109/56.769 (cit. on pp. 82, 83, 85, 86).
- [88] B.-H. Kim, S.-R. Oh, B.-J. Yi, and I. H. Suh. “Optimal Grasping Based on Non-Dimensionalized Performance Indices”. In: *Proceedings 2001 IEEE/RSJ International Conference on Intelligent Robots and Systems. Expanding the Societal Role of Robotics in the the Next Millennium (Cat. No.01CH37180)*. Vol. 2. Oct. 2001, pp. 949–956. DOI: 10.1109/IR0S.2001.976291 (cit. on pp. 82, 83, 86).
- [89] B. Mirtich and J. Canny. “Easily Computable Optimum Grasps in 2-D and 3-D”. In: *Proceedings of the 1994 IEEE International Conference on Robotics and Automation*. Vol. 1. May 1994, pp. 739–747. DOI: 10.1109/ROBOT.1994.351399 (cit. on pp. 83, 84, 86).
- [90] E. Chinellato, R.B. Fisher, A. Morales, and A.P. del Pobil. “Ranking Planar Grasp Configurations for a Three-Finger Hand”. In: *2003 IEEE International Conference on Robotics and Automation (Cat. No.03CH37422)*. Vol. 1. Sept. 2003, pp. 1133–1138. DOI: 10.1109/ROBOT.2003.1241745 (cit. on pp. 83, 84).
- [91] E. Chinellato, A. Morales, R.B. Fisher, and A.P. del Pobil. “Visual Quality Measures for Characterizing Planar Robot Grasps”. In: *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 35.1 (Feb. 2005), pp. 30–41. ISSN: 1558-2442. DOI: 10.1109/TSMCC.2004.840061 (cit. on pp. 83, 84).
- [92] J. Ponce, S. Sullivan, A. Sudsang, J.-D. Boissonnat, and J.-P. Merlet. “On Computing Four-Finger Equilibrium and Force-Closure Grasps of Polyhedral Objects”. In: *The International Journal of Robotics Research* 16.1 (Feb. 1, 1997), pp. 11–35. ISSN: 0278-3649. DOI: 10.1177/027836499701600102 (cit. on pp. 83, 84, 86).
- [93] D. Ding, Y.-H. Lee, and S. Wang. “Computation of 3-D Form-Closure Grasps”. In: *IEEE Transactions on Robotics and Automation* 17.4 (Aug. 2001), pp. 515–522. ISSN: 2374-958X. DOI: 10.1109/70.954765 (cit. on pp. 83, 84).
- [94] J. Cornella and R. Suarez. “Fast and Flexible Determination of Force-Closure Independent Regions to Grasp Polygonal Objects”. In: *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*. Apr. 2005, pp. 766–771. DOI: 10.1109/ROBOT.2005.1570210 (cit. on p. 84).
- [95] Y. Li, Y. Yu, and S. Tsujio. “An Analytical Grasp Planning on given Object with Multifingered Hand”. In: *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No.02CH37292)*. Vol. 4. May 2002, pp. 3749–3754. DOI: 10.1109/ROBOT.2002.1014296 (cit. on pp. 84, 86).
- [96] V.-D. Nguyen. “Constructing Force- Closure Grasps”. In: *The International Journal of Robotics Research* 7.3 (June 1, 1988), pp. 3–16. ISSN: 0278-3649. DOI: 10.1177/027836498800700301 (cit. on p. 84).

- [97] J. Ponce and B. Faverjon. “On Computing Three-Finger Force-Closure Grasps of Polygonal Objects”. In: *IEEE Transactions on Robotics and Automation* 11.6 (Dec. 1995), pp. 868–881. ISSN: 2374-958X. DOI: 10.1109/70.478433 (cit. on pp. 84, 86).
- [98] B. Mishra and M. Teichmann. “The power of friction: Quantifying the "goodness" of frictional grasps”. In: *Algorithms for robotic motion and manipulation*. AK Peters, 1997, pp. 311–320 (cit. on pp. 85, 86).
- [99] A. Caldas et al. “New metric for wrench space reachability of multifingered hand with contact uncertainties”. In: *2014 IEEE/ASME International Conference on Advanced Intelligent Mechatronics*. Besacon: IEEE, July 2014, pp. 1236–1242. ISBN: 978-1-4799-5736-1. DOI: 10.1109/AIM.2014.6878251 (cit. on pp. 85, 86).
- [100] C. A. Klein and B. E. Blaho. “Dexterity Measures for the Design and Control of Kinematically Redundant Manipulators”. In: *The International Journal of Robotics Research* 6.2 (June 1, 1987), pp. 72–83. ISSN: 0278-3649. DOI: 10.1177/027836498700600206 (cit. on pp. 85, 86).
- [101] Tsuneo Yoshikawa. “Manipulability of Robotic Mechanisms”. In: *The International Journal of Robotics Research* 4.2 (June 1, 1985), pp. 3–9. ISSN: 0278-3649. DOI: 10.1177/027836498500400201 (cit. on pp. 85, 86).
- [102] J. K. Salisbury and J. J. Craig. “Articulated Hands: Force Control and Kinematic Issues”. In: *The International Journal of Robotics Research* 1.1 (Mar. 1, 1982), pp. 4–17. ISSN: 0278-3649. DOI: 10.1177/027836498200100102 (cit. on pp. 85, 86).
- [103] S. L. Chiu. “Task Compatibility of Manipulator Postures”. In: *The International Journal of Robotics Research* 7.5 (Oct. 1, 1988), pp. 13–21. ISSN: 0278-3649. DOI: 10.1177/027836498800700502 (cit. on p. 87).
- [104] H. Mnyussiwalla, P. Seguin, P. Vulliez, and J. P. Gazeau. “Evaluation and selection of grasp quality criteria for dexterous manipulation”. In: *Journal of Intelligent & Robotic Systems* 104.2 (Feb. 2022), p. 20. ISSN: 0921-0296, 1573-0409. DOI: 10.1007/s10846-021-01554-4 (cit. on p. 88).
- [105] K. Pearson. “LIII. On Lines and Planes of Closest Fit to Systems of Points in Space”. In: *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science* 2.11 (Nov. 1, 1901), pp. 559–572. ISSN: 1941-5982. DOI: 10.1080/14786440109462720 (cit. on p. 96).
- [106] W. S. Torgerson. “Multidimensional Scaling: I. Theory and Method”. In: *Psychometrika* 17.4 (Dec. 1952), pp. 401–419. ISSN: 0033-3123, 1860-0980. DOI: 10.1007/BF02288916 (cit. on p. 97).
- [107] B. Schölkopf, A. Smola, and K.-R. Müller. “Kernel principal component analysis”. In: *Artificial Neural Networks — ICANN’97*. Vol. 1327. Series Title: Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, 1997, pp. 583–588. ISBN: 978-3-540-63631-1. DOI: 10.1007/BFb0020217 (cit. on pp. 97, 98, 134).

-
- [108] B. Scholkopf, C. Burges, and V. Vapnik. “Extracting Support Data for a Given Task”. In: (1995), pp. 252–257 (cit. on p. 98).
- [109] B. E. Boser, I. M. Guyon, and V. N. Vapnik. “A Training Algorithm for Optimal Margin Classifiers”. In: *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*. COLT '92. New York, NY, USA: Association for Computing Machinery, July 1, 1992, pp. 144–152. ISBN: 978-0-89791-497-0. DOI: 10.1145/130385.130401 (cit. on p. 98).
- [110] M.A. Ajzerman, Eh.M. Braverman, and L.I. Rozonoehr. “Theoretical Foundations of the Potential Function Method in Pattern Recognition Learning”. In: *Automation and Remote Control* 25.6 (Jan. 1, 1964), pp. 821–837 (cit. on p. 98).
- [111] G. H. Bakır, J. Weston, and B. Schölkopf. “Learning to Find Pre-Images”. In: *Proceedings of the 16th International Conference on Neural Information Processing Systems*. NIPS'03. Cambridge, MA, USA: MIT Press, Dec. 9, 2003, pp. 449–456 (cit. on pp. 98, 134).
- [112] S. T. Roweis and L. K. Saul. “Nonlinear Dimensionality Reduction by Locally Linear Embedding”. In: *Science* (Dec. 22, 2000). DOI: 10.1126/science.290.5500.2323 (cit. on p. 98).
- [113] J. B. Tenenbaum, V. de Silva, and J. C. Langford. “A Global Geometric Framework for Nonlinear Dimensionality Reduction”. In: *Science* (Dec. 22, 2000). DOI: 10.1126/science.290.5500.2319 (cit. on p. 99).
- [114] D. H. Ballard. “Modular Learning in Neural Networks”. In: *Proceedings of the Sixth National Conference on Artificial Intelligence*. Vol. 1. AAAI'87. Seattle, Washington: AAAI Press, July 13, 1987, pp. 279–284. ISBN: 978-0-934613-42-2 (cit. on p. 99).
- [115] Y. Lecun. *PhD Thesis: Modeles Connexionnistes de l'apprentissage (Connectionist Learning Models)*. Universite P. et M. Curie (Paris 6), June 1987 (cit. on p. 99).
- [116] H. Bourlard and Y. Kamp. “Auto-Association by Multilayer Perceptrons and Singular Value Decomposition”. In: *Biological Cybernetics* 59.4 (Sept. 1, 1988), pp. 291–294. ISSN: 1432-0770. DOI: 10.1007/BF00332918 (cit. on p. 99).
- [117] G. E. Hinton and R. S. Zemel. “Autoencoders, Minimum Description Length and Helmholtz Free Energy”. In: *Proceedings of the 6th International Conference on Neural Information Processing Systems*. NIPS'93. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., Nov. 29, 1993, pp. 3–10 (cit. on p. 99).
- [118] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. <http://www.deeplearningbook.org>. MIT Press, 2016 (cit. on pp. 99, 102, 103, 104, 105, 111).

- [119] Y. A. LeCun, L. Bottou, G. B. Orr, and K.-R. Müller. “Efficient BackProp”. In: *Neural Networks: Tricks of the Trade: Second Edition*. Ed. by G. Montavon, G. B. Orr, and K.-R. Müller. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2012, pp. 9–48. ISBN: 978-3-642-35289-8. DOI: 10.1007/978-3-642-35289-8_3 (cit. on p. 100).
- [120] X. Glorot, A. Bordes, and Y. Bengio. “Deep Sparse Rectifier Neural Networks”. In: *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*. JMLR Workshop and Conference Proceedings, June 14, 2011, pp. 315–323 (cit. on p. 100).
- [121] Augustin Cauchy et al. “Méthode générale pour la résolution des systemes d’équations simultanées”. In: *Comp. Rend. Sci. Paris* 25.1847 (1847), pp. 536–538 (cit. on p. 101).
- [122] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. “Learning Representations by Back-Propagating Errors”. In: *Nature* 323.6088 (6088 Oct. 1986), pp. 533–536. ISSN: 1476-4687. DOI: 10.1038/323533a0 (cit. on p. 101).
- [123] M. A. Nielsen. *Neural Networks and Deep Learning*. Determination Press, 2015 (cit. on p. 102).
- [124] S. Ruder. *An Overview of Gradient Descent Optimization Algorithms*. June 15, 2017. arXiv: 1609.04747 [cs]. URL: <http://arxiv.org/abs/1609.04747> (visited on Nov. 25, 2021) (cit. on p. 102).
- [125] J. Kiefer and J. Wolfowitz. “Stochastic Estimation of the Maximum of a Regression Function”. In: *The Annals of Mathematical Statistics* 23.3 (1952), pp. 462–466. ISSN: 0003-4851. JSTOR: 2236690 (cit. on p. 102).
- [126] G. Hinton, N. Srivastava, and K. Swersky. *Neural Networks for Machine Learning, Lecture 6a, Overview of mini-batch gradient descent*. URL: http://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf (visited on Nov. 25, 2021) (cit. on p. 102).
- [127] D. P. Kingma and J. Ba. “Adam: A Method for Stochastic Optimization”. In: *3rd International Conference on Learning Representations, ICLR 2015*. May 2015. arXiv: 1412.6980 (cit. on p. 102).
- [128] K. Hornik, M. Stinchcombe, and H. White. “Multilayer Feedforward Networks Are Universal Approximators”. In: *Neural Networks* 2.5 (Jan. 1, 1989), pp. 359–366. ISSN: 0893-6080. DOI: 10.1016/0893-6080(89)90020-8 (cit. on p. 102).
- [129] G. Cybenko. “Approximation by Superpositions of a Sigmoidal Function”. In: *Mathematics of Control, Signals and Systems* 2.4 (Dec. 1, 1989), pp. 303–314. ISSN: 1435-568X. DOI: 10.1007/BF02551274 (cit. on p. 102).

-
- [130] Y. LeCun, P. Haffner, L. Bottou, and Y. Bengio. “Object Recognition with Gradient-Based Learning”. In: *Shape, Contour and Grouping in Computer Vision*. Ed. by D. A. Forsyth, J. L. Mundy, V. di Gesú, and R. Cipolla. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 1999, pp. 319–345. ISBN: 978-3-540-46805-9. DOI: 10.1007/3-540-46805-6_19 (cit. on pp. 103, 150).
- [131] C. G. Turhan and H. S. Bilge. “Recent Trends in Deep Generative Models: A Review”. In: *2018 3rd International Conference on Computer Science and Engineering (UBMK)*. Sept. 2018, pp. 574–579. DOI: 10.1109/UBMK.2018.8566353 (cit. on pp. 104, 106).
- [132] R. Salakhutdinov, A. Mnih, and G. Hinton. “Restricted Boltzmann Machines for Collaborative Filtering”. In: *Proceedings of the 24th International Conference on Machine Learning*. ICML ’07. New York, NY, USA: Association for Computing Machinery, June 20, 2007, pp. 791–798. ISBN: 978-1-59593-793-3. DOI: 10.1145/1273496.1273596 (cit. on p. 104).
- [133] R. Salakhutdinov and G. Hinton. “Deep Boltzmann Machines”. In: *Proceedings of the Twelfth International Conference on Artificial Intelligence and Statistics*. PMLR, Apr. 15, 2009, pp. 448–455 (cit. on p. 104).
- [134] G. E. Hinton, S. Osindero, and Y.-W. Teh. “A Fast Learning Algorithm for Deep Belief Nets”. In: *Neural Computation* 18.7 (July 1, 2006), pp. 1527–1554. ISSN: 0899-7667. DOI: 10.1162/neco.2006.18.7.1527 (cit. on p. 104).
- [135] Diederik Kingma and Max Welling. “Auto-Encoding Variational Bayes”. In: *International Conference on Learning Representations (ICLR)*. 2014 (cit. on pp. 104, 107, 109, 110).
- [136] I. Goodfellow et al. “Generative Adversarial Nets”. In: *Advances in Neural Information Processing Systems*. Vol. 27. Curran Associates, Inc., 2014 (cit. on pp. 105, 106).
- [137] C. Szegedy et al. *Intriguing Properties of Neural Networks*. Feb. 19, 2014. arXiv: 1312.6199 [cs]. URL: <http://arxiv.org/abs/1312.6199> (visited on Dec. 1, 2021) (cit. on p. 106).
- [138] A. Vaswani et al. “Attention Is All You Need”. In: *Advances in Neural Information Processing Systems*. Vol. 30. Curran Associates, Inc., 2017 (cit. on p. 106).
- [139] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. *Neural Machine Translation by Jointly Learning to Align and Translate*. May 19, 2016. arXiv: 1409.0473 [cs, stat]. URL: <http://arxiv.org/abs/1409.0473> (visited on Dec. 1, 2021) (cit. on p. 106).
- [140] Alexey Dosovitskiy et al. “An Image Is Worth 16x16 Words: Transformers for Image Recognition at Scale”. In: Sept. 28, 2020 (cit. on p. 106).

- [141] M. Padala, D. Das, and S. Gujar. “Effect of Input Noise Dimension in GANs”. In: *Neural Information Processing*. Ed. by T. Mantoro, M. Lee, M. A. Ayu, K. W. Wong, and A. N. Hidayanto. Cham: Springer International Publishing, 2021, pp. 558–569. ISBN: 978-3-030-92238-2. DOI: 10.1007/978-3-030-92238-2_46 (cit. on p. 107).
- [142] C. Doersch. *Tutorial on Variational Autoencoders*. Jan. 3, 2021. arXiv: 1606.05908 [cs, stat]. URL: <http://arxiv.org/abs/1606.05908> (visited on Nov. 26, 2021) (cit. on pp. 107, 109, 112).
- [143] L. Devroye. “Sample-Based Non-Uniform Random Variate Generation”. In: *Proceedings of the 18th Conference on Winter Simulation*. WSC ’86. New York, NY, USA: Association for Computing Machinery, Dec. 1, 1986, pp. 260–265. ISBN: 978-0-911801-11-8. DOI: 10.1145/318242.318443 (cit. on p. 108).
- [144] S. Kullback and R. A. Leibler. “On Information and Sufficiency”. In: *The Annals of Mathematical Statistics* 22.1 (1951), pp. 79–86. DOI: 10.1214/aoms/1177729694 (cit. on p. 109).
- [145] I. Higgins et al. “beta-VAE: Learning Basic Visual Concepts with a Constrained Variational Framework”. In: *International Conference on Learning Representation*. 2017 (cit. on pp. 112, 132, 133).
- [146] C. P. Burgess et al. “Understanding disentangling in β -VAE”. In: *arXiv:1804.03599 [cs, stat]* (Apr. 10, 2018). arXiv: 1804.03599 (cit. on pp. 112, 164, 165).
- [147] K. Sohn, X. Yan, and H. Lee. “Learning Structured Output Representation Using Deep Conditional Generative Models”. In: *International Conference on Neural Information Processing Systems*. Vol. 2. NIPS’15. Montreal, Canada: MIT Press, 2015, pp. 3483–3491 (cit. on pp. 113, 128, 166).
- [148] Tridimeo. *3D vision solution for robotic guidance with Tridimeo*. URL: www.tridimeo.com (cit. on p. 123).
- [149] F. Mayran de Chamisso, M. Tamaazousti, and B. Meden. “Method, computer program and system for object detection and location in a three-dimensional scene”. Pat. WO2020065177 (international). CEA. Apr. 2, 2020 (cit. on p. 123).
- [150] F. Mayran de Chamisso, M. Tamaazousti, and B. Meden. “Method, computer program and system for identifying an object instance in a three-dimensional scene”. Pat. WO2020201392 (international). CEA. Oct. 9, 2020 (cit. on p. 123).
- [151] M. Quigley et al. “ROS: an open-source Robot Operating System”. In: *ICRA workshop on open source software*. Vol. 3. 3.2. Kobe, Japan. 2009, p. 5 (cit. on p. 124).
- [152] N. Koenig and A. Howard. “Design and use paradigms for Gazebo, an open-source multi-robot simulator”. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Vol. 3. 2004, pp. 2149–2154. DOI: 10.1109/IROS.2004.1389727 (cit. on p. 124).

-
- [153] Sachin Chitta et al. “ros_control: A generic and simple control framework for ROS”. In: *The Journal of Open Source Software* (2017). DOI: 10.21105/joss.00456 (cit. on p. 125).
- [154] D. Coleman, I. S Ucan, S. Chitta, and N. Correll. “Reducing the Barrier to Entry of Complex Robotic Software: A MoveIt! Case Study”. In: *Journal of Software Engineering for Robotics* 5.1 (2014), p. 14 (cit. on pp. 125, 175, 181).
- [155] François Chollet et al. *Keras*. Software available from www.keras.io. 2015 (cit. on pp. 127, 128, 166).
- [156] Martín Abadi et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from www.tensorflow.org. 2015 (cit. on pp. 128, 166).
- [157] F. Pedregosa et al. “Scikit-learn: Machine Learning in Python”. In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830 (cit. on pp. 134, 152).
- [158] C.M. Hoffmann and J.R. Rossignac. “A Road Map to Solid Modeling”. In: *IEEE Transactions on Visualization and Computer Graphics* 2.1 (Mar. 1996), pp. 3–10. ISSN: 1941-0506. DOI: 10.1109/2945.489381 (cit. on p. 146).
- [159] C. D. Toth, J. O’Rourke, and J. E. Goodman. *Handbook of Discrete and Computational Geometry*. CRC Press, Nov. 22, 2017. 1951 pp. ISBN: 978-1-4987-1142-5 (cit. on p. 146).
- [160] User Captain Sprite on en.wikipedia. *Constructive Solid Geometry*. URL: https://en.wikipedia.org/wiki/Constructive_solid_geometry (visited on Dec. 15, 2021) (cit. on p. 146).
- [161] H. B. Voelcker and A. A. G. Requicha. *Constructive Solid Geometry*. Technical Report 25. Rochester, N.Y.: University of Rochester. Production Automation Project, 1977. 46 pp. (cit. on p. 147).
- [162] I. C. Braid. “The Synthesis of Solids Bounded by Many Faces”. In: *Communications of the ACM* 18.4 (Apr. 1, 1975), pp. 209–216. ISSN: 0001-0782. DOI: 10.1145/360715.360727 (cit. on p. 147).
- [163] C. Kingkan and K. Hashimoto. “Generating Mesh-based Shapes From Learned Latent Spaces of Point Clouds with VAE-GAN”. In: *2018 24th International Conference on Pattern Recognition (ICPR)*. Aug. 2018, pp. 308–313. DOI: 10.1109/ICPR.2018.8546232 (cit. on pp. 150, 156).
- [164] C. R. Qi, H. Su, K. Mo, and L. J. Guibas. “PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 652–660 (cit. on p. 150).
- [165] P. Achlioptas, O. Diamanti, I. Mitliagkas, and L. Guibas. “Learning Representations and Generative Models for 3D Point Clouds”. In: *Proceedings of the 35th International Conference on Machine Learning*. PMLR, July 3, 2018, pp. 40–49 (cit. on pp. 150, 156).

- [166] I. Kostrikov, Z. Jiang, D. Panozzo, D. Zorin, and J. Bruna. “Surface Networks”. In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. June 2018, pp. 2540–2548. DOI: 10.1109/CVPR.2018.00269 (cit. on pp. 150, 156).
- [167] D. George, X. Xie, and G. KL Tam. “3D Mesh Segmentation via Multi-Branch 1D Convolutional Neural Networks”. In: *Graphical Models* 96 (Mar. 1, 2018), pp. 1–10. ISSN: 1524-0703. DOI: 10.1016/j.gmod.2018.01.001 (cit. on p. 150).
- [168] P. Wang et al. “3D Shape Segmentation via Shape Fully Convolutional Networks”. In: *Computers & Graphics* 76 (Nov. 1, 2018), pp. 182–192. ISSN: 0097-8493. DOI: 10.1016/j.cag.2018.07.011 (cit. on p. 150).
- [169] A. Ranjan, T. Bolkart, S. Sanyal, and M. J. Black. “Generating 3D Faces Using Convolutional Mesh Autoencoders”. In: *Computer Vision – ECCV 2018*. Ed. by V. Ferrari, M. Hebert, C. Sminchisescu, and Y. Weiss. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2018, pp. 725–741. ISBN: 978-3-030-01219-9. DOI: 10.1007/978-3-030-01219-9_43 (cit. on pp. 150, 156).
- [170] Y.-L. Qiao et al. “Learning on 3D Meshes with Laplacian Encoding and Pooling”. In: *IEEE Transactions on Visualization and Computer Graphics* (2020), pp. 1–1. ISSN: 1941-0506. DOI: 10.1109/TVCG.2020.3014449 (cit. on p. 150).
- [171] V. V. Singh, S. V. Sheshappanavar, and C. Kambhamettu. “Mesh Classification With Dilated Mesh Convolutions”. In: *2021 IEEE International Conference on Image Processing (ICIP)*. Sept. 2021, pp. 3138–3142. DOI: 10.1109/ICIP42928.2021.9506311 (cit. on p. 150).
- [172] Y.-J. Yuan et al. “Mesh Variational Autoencoders with Edge Contraction Pooling”. In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. June 2020, pp. 1105–1112. DOI: 10.1109/CVPRW50498.2020.00145 (cit. on pp. 150, 156).
- [173] Q. Tan, L. Gao, Y.-K. Lai, and S. Xia. “Variational Autoencoders for Deforming 3D Mesh Models”. In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. June 2018, pp. 5841–5850. DOI: 10.1109/CVPR.2018.00612 (cit. on pp. 150, 156).
- [174] A. Brock, T. Lim, J. M. Ritchie, and N. Weston. *Generative and Discriminative Voxel Modeling with Convolutional Neural Networks*. Aug. 16, 2016. arXiv: 1608.04236 [cs, stat]. URL: <http://arxiv.org/abs/1608.04236> (visited on Dec. 17, 2021) (cit. on pp. 150, 156).
- [175] D. Maturana and S. Scherer. “VoxNet: A 3D Convolutional Neural Network for Real-Time Object Recognition”. In: *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Sept. 2015, pp. 922–928. DOI: 10.1109/IROS.2015.7353481 (cit. on p. 150).
- [176] N. Sedaghat, M. Zolfaghari, E. Amiri, and T. Brox. “Orientation-Boosted Voxel Nets for 3D Object Recognition”. In: *Proceedings of the British Machine Vision Conference 2017*. London, UK: British Machine Vision Association, 2017, p. 97. ISBN: 978-1-901725-60-5. DOI: 10.5244/C.31.97 (cit. on p. 150).

-
- [177] M. Defferrard, X. Bresson, and P. Vandergheynst. “Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering”. In: *Proceedings of the 30th International Conference on Neural Information Processing Systems*. NIPS’16. Red Hook, NY, USA: Curran Associates Inc., Dec. 5, 2016, pp. 3844–3852. ISBN: 978-1-5108-3881-9 (cit. on p. 150).
- [178] M. Henaff, J. Bruna, and Y. LeCun. *Deep Convolutional Networks on Graph-Structured Data*. June 16, 2015. arXiv: 1506.05163 [cs]. URL: <http://arxiv.org/abs/1506.05163> (visited on Dec. 13, 2021) (cit. on p. 150).
- [179] T. N. Kipf and M. Welling. “Semi-Supervised Classification with Graph Convolutional Networks”. In: *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017 (cit. on p. 150).
- [180] J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun. *Spectral Networks and Locally Connected Networks on Graphs*. May 21, 2014. arXiv: 1312.6203 [cs]. URL: <http://arxiv.org/abs/1312.6203> (visited on Dec. 13, 2021) (cit. on p. 150).
- [181] M. M. Bronstein, J. Bruna, Y. LeCun, A. Szlam, and P. Vandergheynst. “Geometric Deep Learning: Going beyond Euclidean Data”. In: *IEEE Signal Processing Magazine* 34.4 (July 2017), pp. 18–42. ISSN: 1558-0792. DOI: 10.1109/MSP.2017.2693418 (cit. on p. 150).
- [182] Angel X. Chang et al. *ShapeNet: An Information-Rich 3D Model Repository*. Tech. rep. arXiv:1512.03012 [cs.GR]. Stanford University — Princeton University — Toyota Technological Institute at Chicago, 2015 (cit. on p. 150).
- [183] U. von Luxburg. “A Tutorial on Spectral Clustering”. In: *Statistics and Computing* 17.4 (Dec. 2007), pp. 395–416. ISSN: 0960-3174, 1573-1375. DOI: 10.1007/s11222-007-9033-z (cit. on p. 151).
- [184] M. Meyer, M. Desbrun, P. Schröder, and A. H. Barr. “Discrete Differential-Geometry Operators for Triangulated 2-Manifolds”. In: *Visualization and Mathematics III*. Ed. by H.-C. Hege and K. Polthier. Mathematics and Visualization. Berlin, Heidelberg: Springer, 2003, pp. 35–57. ISBN: 978-3-662-05105-4. DOI: 10.1007/978-3-662-05105-4_2 (cit. on p. 152).
- [185] N. Sharp and K. Crane. “A Laplacian for Nonmanifold Triangle Meshes”. In: *Computer Graphics Forum* 39.5 (Aug. 2020), pp. 69–80. ISSN: 0167-7055, 1467-8659. DOI: 10.1111/cgf.14069 (cit. on p. 152).
- [186] A. L. Maas, A. Y. Hannun, and A. Y. Ng. “Rectifier Nonlinearities Improve Neural Network Acoustic Models”. In: *(ICML) International Conference on Machine Learning. Workshop on Deep Learning for Audio, Speech and Language Processing*. June 13, 2013, p. 6 (cit. on p. 156).
- [187] Blender Online Community. *Blender - a 3D modelling and rendering package*. Blender Foundation. 2018. URL: <http://www.blender.org> (cit. on p. 158).

- [188] H. Li, Z. Xu, G. Taylor, C. Studer, and T. Goldstein. “Visualizing the loss landscape of neural nets”. In: *Proceedings of the 32nd International Conference on Neural Information Processing Systems*. NIPS’18. Red Hook, NY, USA: Curran Associates Inc., Dec. 2018, pp. 6391–6401 (cit. on p. 162).
- [189] NASA Goddard Photo and Video. *Exploring the Universe in Virtual Reality*. URL: https://commons.wikimedia.org/wiki/File:Exploring_the_Universe_in_Virtual_Reality.jpg (cit. on p. 175).
- [190] R. Detry, J. Papon, and L. Matthies. “Task-oriented grasping with semantic and geometric scene understanding”. In: *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. ISSN: 2153-0866. Sept. 2017, pp. 3266–3273. DOI: 10.1109/IROS.2017.8206162 (cit. on p. 177).
- [191] A. Sahbani and S. El-Khoury. “A hybrid approach for grasping 3D objects”. In: *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*. ISSN: 2153-0866. Oct. 2009, pp. 1272–1277. DOI: 10.1109/IROS.2009.5354105 (cit. on p. 177).
- [192] U. Vollhardt, M. Makarov, A. Caldas, M. Grossard, and P. Rodriguez-Ayerbe. “Energy-based stability analysis for grasp selection with compliant multi-fingered hands”. In: *2019 18th European Control Conference (ECC)*. June 2019, pp. 1592–1597. DOI: 10.23919/ECC.2019.8795792 (cit. on p. 177).
- [193] R. Haschke, J.J. Steil, I. Steuwer, and H. Ritter. “Task-oriented quality measures for dextrous grasping”. In: *2005 International Symposium on Computational Intelligence in Robotics and Automation*. June 2005, pp. 689–694. DOI: 10.1109/CIRA.2005.1554357 (cit. on p. 177).
- [194] G. Borgefors. “Distance transformations in arbitrary dimensions”. In: *Computer Vision, Graphics, and Image Processing 27.3* (Sept. 1, 1984), pp. 321–345. ISSN: 0734-189X. DOI: 10.1016/0734-189X(84)90035-5 (cit. on p. 179).
- [195] Kinova robotics. *Gain autonomy*. URL: <https://assistive.kinovarobotics.com/> (cit. on p. 179).

