POLITECNICO DI MILANO

Facoltà di Ingegneria
Scuola di Ingegneria Industriale e dell'Informazione
Dipartimento di Elettronica, Informazione e Bioingegneria

Master of Science in
Computer Science and Engineering

Affordance Prediction with Vision via Task Oriented Grasp Quality Metrics

Advisor: PROF. MATTEO MATTEUCCI

Co-advisor: PROF. MARINA INDRI

Master Graduation Thesis by:

LUCA CAVALLI
Student Id n. 893683

Academic Year 2018-2019

ACKNOWLEDGMENTS

I sincerely thank my friend and colleague Gianpaolo Di Pietro for collaborating in the implementation of this research work together with its early publication, as well as I want to thank my advisor Matteo Matteucci for his precious directions in elaborating the research line followed in this thesis.

I want to thank my family for their moral and economic support that allowed me to concentrate full time on this phase of my studies, culminating in this work.

Finally, I am grateful to all the great people that I knew in these five years at Politecnico Di Milano and Politecnico di Torino, whose interaction proved extremely enriching both from an academic and from a personal point of view. Among these I want to reserve special thanks to my friend Gaia, as her moral support led to more than one successful decision that finally contributed to arriving to this thesis.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACRONYMS

ICRA	International Conference on Robotics and Automation
IROS	International Conference on Intelligent Robots and System
RSS	Robotics: Science and Systems
IWCMAR	International Workshop on Computational Models of Affordances in Robotics
GGS	Good Grasp Samples. Grasp samples which respect the condition in Section 6.1 and differ by grasp or use location
UGG	Unique Good Grasps. Grasp samples which respect the condition in Section 6.1 and differ by grasp
CGAL	Computational Geometry Algorithms Library
PSB	Princeton Shape Benchmark
LP	Linear Programming
GUI	Graphical User Interface
RAM	Random Access Memory
DoF	Degree of Freedom
MSE	Mean Squared Error
CA	Comparison Accuracy
GMS	Global Minimum Score
CNN	Convolutional Neural Network
PN	PointNet
MLP	Multi-Layer Perceptron
ReLU	Rectified Linear Unit
iid	independent identically distributed

ABSTRACT

The research community has spent much effort in tackling the problem of grasping novel objects in different settings [30, 31, 37, 61, 62] with the objective of holding objects robustly with robotic manipulators; however, real manipulation tasks go far beyond holding the objects, and the quality of a grasp depends on the task it is meant to support. While many quality metrics exist to evaluate the quality of a grasp by itself [9, 48], no clear quantification of the quality of a grasp relatively to a task has been defined. In this thesis we propose a framework to extend the concept of quality metric to task-oriented grasping by defining general physical measures for an open set of task affordances. We evaluate both the results provided by such metrics and their applicability in practice by learning to infer them from vision. To validate our framework, we have collected a dataset of grasps and computed elementary grap metrics by using the GraspIt! simulator [18] on the Princeton Shape Benchmark object models [19]. Then we have trained deep models to infer these elementary metrics from range images taken in a simulated camera-in-hand setting to assess the applicability of our framework to more realistic partial-information settings. Experimental results show that a direct search in a perfect information simulation with our novel framework produces new and creative grasps which fit the specific actuator in use for the selected task, while inference of such metrics from synthesized vision provides already meaningful results, leaving margins for further improvement.

SOMMARIO

La comunità di ricerca ha speso molte energie per affrontare il problema di afferrare oggetti non noti a priori in diversi contesti [30, 31, 37, 61, 62] con l'obiettivo di generare prese robuste con manipolatori robotici; tuttavia le applicazioni reali vanno molto oltre il tenere semplicemente gli oggetti, e la qualità di una presa dipende dal compito di manipolazione che la stessa deve supportare. Esistono molte metriche per valutare la qualità di una presa a sé stante [9, 48], tuttavia non esiste una chiara quantificazione della qualità di una presa relativamente ad un compito di manipolazione. In questa tesi proponiamo una metodologia per estendere il concetto di metrica alle prese orientate ad un compito definendo misure fisiche generali per un insieme aperto di possibili compiti. Valutiamo sia la validità di queste metriche che la loro applicabilità pratica apprendendo l'inferenza delle stesse dalla visione.

INDIVIDUAZIONE DEL PROBLEMA

Una caratteristica molto distintiva dei primati sono le mani. Queste permettono una manipolazione agile degli oggetti trovati nell'ambiente circostante e, con ragionevoli limitazioni, garantiscono questa abilità indipendentemente da molte caratteristiche dell'oggetto come la forma, le dimensioni e la consistenza. Poiché questa caratteristica rende le mani degli attuatori estremamente generali e versatili per interagire con l'ambiente, oggi le imitiamo nei robot, costruendo mani robotiche.

Tuttavia usare davvero delle mani per interagire con l'ambiente è un problema estremamente complesso. Ad oggi la maggiore applicazione di mani e bracci robotici è nell'automazione delle linee di assemblaggio, non per caso: le linee di assemblaggio spesso presentano un grado di incertezza basso o nullo nell'ambiente di lavoro, e richiedono azioni ripetitive che non necessitano di una pianificazione della presa sul momento. Molti bracci robotici di successo nel mercato delle linee di assemblaggio sono infatti progettati per eseguire dei movimenti fissi con tempi e posizionamenti molto precisi, azione che richiede un controllo meccanico molto elaborato (un altro problema molto complesso che non consideriamo in questa tesi), ma nessuna pianificazione, ponendosi quindi molto lontano dall'attuatore generale rappresentato dalle mani dei primati.

Più recentemente, con gli ultimi avanzamenti nel campo della Computer Vision, si è riusciti a raggiungere una pianificazione della presa, gestendo più o meno limitati gradi di incertezza nell'ambiente, ma ancora delle applicazioni di successo nel mercato in cui c'è incertezza sulla geometria stes-

sa degli oggetti sono estremamente rare a causa della difficoltà di questo problema. Esempi di applicazioni, come nella gestione dei rifiuti [42], sono ancora limitate al riposizionamento di oggetti e sono ancora molto lontane dall'afferrare oggetti per manipolarli come strumenti.

Sbloccare tutte le potenzialità delle mani robotiche permetterebbe un'interazione molto più ricca dei robot autonomi con l'ambiente circostante. Per arrivare a questo, abbiamo bisogno di pianificare prese sugli oggetti in funzione di ciò che dobbiamo fare con quesi oggetti; questo problema va sotto il nome di *Task Oriented Grasping*. Il Task Oriented Grasping è un campo di ricerca molto recente e molto attivo da cui, al meglio della nostra conoscenza, non è scaturito ancora nessun esempio maturo di applicazione di mercato ed ha al contempo il potenziale di essere applicato in molte situazioni.

In questa tesi ci concentreremo sull'applicazione del braccio robotico come assistente di cucina, che deve essere in grado di usare strumenti di cucina per compiere compiti comuni e insoliti con gli oggetti disponibili che può raggiungere. Esempi di compiti in questo contesto sono spostare oggetti, tagliare, versare, spingere, tirare e battere. Inoltre, non ammettiamo nessuna ipotesi sulla natura degli strumenti a disposizione, che potrebbero essere di qualsiasi forma e dimensione.

CONTRIBUTI

Con questa tesi portiamo un contributo di ricerca nel campo della Percezione dell'Affordance per il Task Oriented Grasping.

Nel nostro contributo:

1. Definiamo e validiamo qualitativamente un framework per valutare la qualità di una presa orientata ad un task
2. Produciamo un plugin per GraspIt! [18] per generare dati annotati con minima o nulla necessità di intervento umano, dunque in modo estremamente scalabile
3. Generiamo due dataset distinti di 400M e 100M di prese annotate su 22 oggetti dal Princeton Shape Benchmark [19] che pianifichiamo di rendere pubblici nel futuro prossimo
4. Proponiamo e valutiamo quantitativamente e qualitativamente dei modelli per affrontare il problema di imparare a predire dalla visione le nostre metriche di qualità orientate a task
5. Proponiamo un nuovo modello, che chiamiamo Local PointNet, che mette insieme l'idea della PointNet [56] di costruire embeddings dei punti da una nuvola di punti con l'elemento convoluzionale delle Convolutional Neural Networks che modella efficacemente le correlazioni

locali e gerarchiche negli input, e mostriamo che supera tutti gli altri modelli provati in alcuni dei nostri problemi di learning

I risultati sperimentali mostrano che una ricerca diretta in un contesto di informazione completa in simulazione con il nostro nuovo framework produce prese nuove e creative che si adattano bene alla specifica mano robotica in uso e al compito selezionato, mentre l'inferenza delle metriche proposte da visione sintetica produce già risultati significativi, lasciando margini per ulteriori miglioramenti.

Una versione preliminare di questa ricerca è stata proposta ed accettata per la poster session del Second International Workshop on Computational Models of Affordances in Robotics (IWCMAR), alla International Conference on Robotics and Automation (ICRA) tenutasi a Maggio 2019, ed è allegata in questa tesi nell'Appendice A.

METODOLOGIA E STRUTTURA DEL DOCUMENTO

Questa tesi è strutturata come un lavoro di ricerca nel contesto dell'*Honours Programme Scientific Research in Information Technology*; ogni Capitolo descrive un passo fondamentale verso la definizione, formulazione e valutazione di questa ricerca.

1. **Definizione del Problema**: nel Capitolo 1 (da cui è tratto questo estratto) formuliamo la definizione del problema, esponendo le ragioni ed il metodo per ricercare in questo campo.

2. **Stato dell'Arte**: nel Capitolo 2 definiamo meglio la specifica area di ricerca e selezioniamo le direzioni di ricerca più promettenti analizzando le limitazioni dello stato dell'arte corrente. Questa analisi è divisa in due fasi, descrivendo prima il contesto della ricerca della robotica autonoma alla larga, e successivamente focalizzando l'analisi sul settore specifico che è pertinente al problema che trattiamo.

3. **Prerequisiti Principali**: nel Capitolo 3 descriviamo i principali modelli per l'elaborazione di immagini e di nuvole di punti che saranno utilizzati per implementare il sistema descritto.

4. **Formulazione della Soluzione**: nel Capitolo 4 proponiamo e formalizziamo la nostra soluzione mantenendo una prospettiva teorica. Fin da questa fase consideriamo il problema dell'applicabilità della soluzione in un contesto reale, pur mantenendo il trattato teorico.

5. **Generazione dei dati**: nel Capitolo 5 descriviamo il processo di raccolta dei dati producendo un plugin GraspIt! per la generazione dei dati come implementazione del sistema proposto nel Capitolo 4.

6. **Validazione Sperimentale**: nel Capitolo 6 descriviamo gli esperimenti condotti con i dati generati per valutare la soluzione proposta nel Capitolo 4 e discuterne i risultati.

7. **Conclusioni**: nel Capitolo 7 discutiamo i risultati e sottolineiamo le limitazioni del lavoro corrente, tracciando possibili linee di future estensioni.

8. **Review**: nell'Appendice A alleghiamo il poster paper accettato e presentato nel Second International Workshop on Computational Models of Affordances in Robotics ad ICRA 2019.

1 INTRODUCTION

The research community has spent much effort in tackling the problem of grasping novel objects in different settings [30, 31, 37, 61, 62] with the objective of holding objects robustly with robotic manipulators; however, real manipulation tasks go far beyond holding the objects, and the quality of a grasp depends on the task it is meant to support. While many quality metrics exist to evaluate the quality of a grasp by itself [9, 48], no clear quantification of the quality of a grasp relatively to a task has been defined. In this thesis we propose a framework to extend the concept of quality metric to task-oriented grasping by defining general physical measures for an open set of task affordances. We evaluate both the results provided by such metrics and their applicability in practice by learning to infer them from vision.

1.1 PROBLEM STATEMENT

One very distinctive feature of primates is hands. Hands allow for dexterous manipulation of objects taken from the environment and, within reasonable limitations, they grant this ability regardless of many object properties such as shape, size and softness. As this feature makes hands an extremely general and versatile actuator to interact with the environment, nowadays we imitate them in robots, building robotic hands.

However, actually using hands to interact with the environment is an extremely complex problem. Today the most important application of robotic hands and arms is in assembly line automation, not by chance: assembly lines usually have a very low degree of uncertainty in the working environment, requiring repetitive actions that do not need online planning of the grasp. Many successful robotic arms for assembly lines are indeed instructed to perform some fixed movements with perfect timing and positioning, which requires very fine mechanical control (another very complex problem that we do not consider in this thesis), but no planning at all, thus being very far from the general actuator that primate hands represent.

More recently, with advances in Computer Vision, we were able to achieve grasp planning, allowing for some degrees of uncertainty in the target environment, but still successful market applications in which there is uncertainty on the shape of the grasped objects are extremely rare due to the difficulty of the problem. Example applications such as in trash management [42] are still limited to the pick-and-place task and are very far from tool manipulation.

Unlocking the full potential of robotic hands would allow a much richer interaction of autonomous robots with the environment. In order to achieve this, we need to be able to plan grasps on objects as a function of what we need to do with such objects, which goes under the name of *Task Oriented Grasping*. Task Oriented Grasping is an active and very recent research field which, to the best of our knowledge, has no mature market example and the potential to be applied in many situations.

In this thesis we focus on the target application of the kitchen assistant robotic arm, which should be able to use kitchen tools to perform common and uncommon kitchen tasks with the available objects to reach. Sample tasks in this context can be pick-and-place, cutting, pouring, pulling, pushing and beating. Moreover, no hypothesis is allowed on the available tools, which may be of any size and shape.

1.2 CONTRIBUTIONS

With this thesis we provide a research contribution within the field of Affordance Perception for Task Oriented Grasping.

In our contribution:

1. We define and qualitatively validate a framework for quality assessment of task-oriented grasps
2. We provide a GraspIt! [18] plugin to produce labelled data with minimal to no human intervention, thus in an extremely scalable way
3. We generated two distinct datasets of 400M and 100M evaluated grasps on 22 objects of the Princeton Shape Benchmark [19] which we plan to make public in the near future
4. We propose, benchmark and qualitatively validate models to tackle the problem of learning to infer our task oriented metrics from vision
5. We propose a novel model, that we call Local PointNet, that merges the PointNet [56] idea of embedding points from a point cloud with the convolutional element of Convolutional Neural Networks that models effectively local and hierarchical correlations between inputs, and we prove that it outperforms all other tested models in some of our learning tasks

Experimental results show that a direct search in a perfect information simulation with our novel framework produces new and creative grasps which fit the specific actuator in use for the selected task, while inference of such metrics from synthesized vision provides already meaningful results, leaving margins for further improvement.

An early version of this work has been proposed and accepted for the poster session of the Second International Workshop on Computational Models of Affordances in Robotics (IWCMAR), at the International Conference on Robotics and Automation (ICRA) held in May 2019, and is included in this thesis in Appendix A.

1.3 METHODOLOGY AND DOCUMENT STRUCTURE

This thesis is structured as a research work within the context of the *Honours Programme Scientific Research in Information Technology*; each Chapter describes a fundamental step towards the definition, formulation and evaluation of this piece of research.

1. **Problem Statement**: in Chapter 1 we formulate the problem statement, giving the reasons for researching in this field.
2. **State of the Art**: in Chapter 2 we better define the specific research field and select the promising research directions by analyzing the limitations of the current State of the Art. This is performed in two steps, by first describing the wide context of the Autonomous Robotics research field and then focusing on the specific sector which is pertinent to our problem statement.
3. **Main Model Prerequisites**: in Chapter 3 we review the main computational models for image and point cloud processing, that are employed to implement the system.
4. **Solution Formulation**: in Chapter 4 we propose and formalize our solution from a theoretical perspective. At this point we already address the problem of applicability of the solution in a real context, but still the problem is treated theoretically.
5. **Data Generation**: in Chapter 5 we describe the process of data collection by building the GraspIt! data generation plugin as an implementation of the system proposed in Chapter 4.
6. **Experimental Evaluation**: in Chapter 6 we describe the experiments conducted with the generated data to evaluate the solution proposed in Chapter 4 and discuss their results.
7. **Conclusions**: in Chapter 7 we discuss the results, and underline the limitations of the current work, drawing possible lines of future works.
8. **Review**: in Appendix A we include the poster paper accepted and presented in the Second International Workshop on Computational Models of Affordances in Robotics at ICRA 2019.

2

GRASPING AND AFFORDANCE BACKGROUND

In this chapter we identify the research fields of Task Oriented Grasping and Affordance Perception. We review and classify the related works to analyse the explored paths and expose the limitations of the current research works to identify open problems.

2.1 ROBOTIC GRASPING

The Robotic Grasping research field tackles the problem of automating grasping actions on novel objects under different sensorimotor conditions; it is tightly connected with the broader field of affordance perception, that we describe in Section 2.2. The core community is in the Robotics research area as main field of application, but connections exist also with the areas of Computer Vision, Machine Learning and Artificial Intelligence in general as they provide fundamental tools to tackle this problem. Analyzing the affordance perception literature we can also find significant connections with the area of Human Computer Interaction with the aim of studying effective human-robot collaboration, with a particular focus on language-based communication.

The challenges and opportunities of our research can be better framed in the more general context of affordance perception, as it gives a more powerful and general high level perspective in planning successive and related actions, introducing to Task Oriented Grasping.

2.1.1 *Basic tools in robotic grasping*

Fundamental tools for grasping come from physics, and they are used to quantitatively evaluate the quality of a grasp given the hand model, its configuration, and contact points. These tools are based on the definition of a *Grasp Matrix* $\mathbf{G}$ and a *Hand Jacobian* $\mathbf{J}$. Let n_c be the number of contact points and n_q be the number of joints in the hand, the Grasp Matrix $\mathbf{G}$ maps the object twists to the transmitted twists in each contact point, while the Hand Jacobian $\mathbf{J}$ maps the joint velocities to the transmitted contact twists on the hand as shown in Equation 2.1.

Let $\boldsymbol{\nu}$ be the twist of the object with respect to a global reference, $\boldsymbol{\nu}_{c,obj}$ be the transmitted twists on the object expressed with respect to each contact point, $\boldsymbol{\nu}_{c,hnd}$ the same on the hand and $\dot{\mathbf{q}}$ the joint velocities, then we have:

$$\begin{aligned} \boldsymbol{\nu}_{c,obj} &= \mathbf{G}^{\mathsf{T}}\boldsymbol{\nu} \\ \boldsymbol{\nu}_{c,hnd} &= \mathbf{J}\dot{\mathbf{q}} \end{aligned} \tag{2.1}$$

The choice of what twists are transmitted between object and hand encodes the contact model; the most used are **Point-contact-without-Friction** (only normal component transmitted, and no momentum), **Hard Finger** (all translational components, no momentum) and **Soft Finger** (all translational components and normal momentum). These matrices only depend on the contact point geometry and hand configuration, and encode all the information about the grasp.

The quantification of the actual closure and robustness of a grasp configuration can be encoded into a linear programming problem, and thus efficiently extracted. We refer to [28] for an in depth analysis on this topic. Many optimization algorithms and simulation tools have been devised around this, in particular we refer to *GraspIt!* which collects a number of evaluation and optimization tools in an open simulated environment [18].

2.2 AFFORDANCE AND TASK ORIENTED GRASPING

Since the first definition of affordances, by Gibson in 1966, [1, 2] a long discussion evolved, of which we consider some main voices, for a complete discussion refer to [60]. We report here the very first definition by Gibson [1], where the affordance is defined as what things furnish to an observer as a consequence of the object properties:

> When the constant properties of constant objects are perceived (the shape, size, color, texture, composition, motion, animation, and position relative to other objects), the observer can go on to detect their affordances. I have coined this word as a substitute for values, a term which carries an old burden of philosophical meaning. I mean simply what things furnish, for good or ill. What they afford the observer, after all, depends on their properties (Gibson 1966, p. 285).

This definition later evolved [2] suggesting that animals directly perceive affordances as peculiar combinations of properties that suggest some possibility of action:

> The psychologists assume that objects are composed of their qualities. But I now suggest that what we perceive when we look

> at objects are their affordances, not their qualities. We can discriminate the dimensions of difference if required to do so in an experiment, but what the object affords us is what we normally pay attention to. The special combination of qualities into which an object can be analyzed is ordinarily not noticed (Gibson 1979, p. 134).

Subsequent authors [3, 5, 7, 12–17] either supported and detailed the main concept by Gibson, underlining the relational nature of affordances as being emergent properties embodied in the relations between an animal and its environment directly connected with the possibility of action of the animal with the environment [3, 14, 16, 17], or suggested some substantially different model like hypothesizing the need for a semantic internal representation of actions in order to conceive affordances, that are thus not directly perceived [5].

Applied to robotics, by affordance perception we mean understanding the possibility of action of a robot depending on the possible relations between its actuators and the environment to achieve high level tasks. In this context grasping represents an affordance for the control of some or all degrees of freedom of some object with a hand-like physical actuator.

Task Oriented Grasping extends the concept of Robotic Grasping by adding a later action to be done with the grasped object. This extension comes with the trivial observation that the same object is better grasped in very different ways depending on what it is planned to be used later for: as an example we can think about how we grasp a hammer far from its center of mass for beating, and how we grasp it near its center of mass for pick-and-place.

In the wider context of affordances, task-oriented grasping enables the possibility of tool use, which in turns allows an enormous range of new possibilities of action (this concept of unlocking new actions through actions goes under the name of *affordance chaining* in the Affordance research community). Being able to model and understand the environment is a critical point to plan a solid grasp, and even more to relate the grasp with a task; for this reason research in Affordance Perception for Robotic Grasping is necessary to achieve general task-oriented grasping.

The recent interest in the field of Affordance Perception is also witnessed by the organization of dedicated workshops; the latest ones include *Learning Object Affordances: a fundamental step to allow prediction, planning and tool use?* at the International Conference on Intelligent Robots and System (IROS) 2015, the *International Workshop on Computational Models of Affordances in Robotics* at Robotics: Science and Systems (RSS) 2018, and the *Second International Workshop on Computational Models of Affordances in Robotics* at the International Conference on Robotics and Automation (ICRA) 2019.

2.3 RELATED WORKS IN ROBOTIC GRASPING

In order to better understand the research landscape around Task Oriented Grasping, we first need to review and classify the research works in the broader field of Robotic Grasping. Such analysis not only describes the broader research context of task-oriented grasping, but it also highlights important research directions that we follow.

2.3.1 *Classification of main related works*

The research community in Robotic Grasping has proposed very different and heterogeneous perceptual models, within very different settings. Focusing only on the perception and modeling of the environment we can notice a strong correlation between the perceptual basis of a model and its limitations. In particular we classify the main related works according to the *dimensionality* of the vision system which models the environment:

- **Blind Perception**: in this category we include all systems which do not employ vision. It is important to underline that here we mean vision in its most general sense, as any perceptual system that can provide a global view of the environment, as a laser system could do. As such, blind systems have only access to local information about the contact points of the fingers, with the consequent strong limitation of not being able to plan any new grasp. The purpose of these works is usually to evaluate or improve an existing grasp to make it more solid via contact information like local pressure maps or motors torque.

- **Appearance Perception**: in this category we include all systems employing at least a monocular vision system or equivalent. They do not necessarily perceive nor model any clue of absolute spacial dimensions and shapes on which to plan grasps, although they have some global perception of the environment.

- **Geometry Perception**: there are many ways a system can estimate real distances and have a notion of three-dimensional space and object shape, ranging from RGB-D cameras to model priors. All works that perceive or model information connected with metric distances about the environment have a clear extra opportunity as they can employ geometrical models to plan accurate grasps. Inside this broad category we can further differentiate according to the span of the spacial model:
  - **Focused**: these works try to estimate or consider a detailed spacial model of the object to be grasped

  - **Global**: these works try to model the whole environment and perceive many objects at once, usually but not necessarily with greater uncertainty than focused models.
  - **Mixed**: these works estimate a model of the environment and also identify distinct object models and their locations

- **Physics Perception**: a further dimension of perception of the environment for its understanding includes the physical perception. Its relevance in understanding the environment comes from the predictive power of physics as a model of the interaction with and between objects. As a consequence, physical quantities efficiently encode the knowledge of the agent about the evolution of the environment in time and as such become a general perceptive model of the "*dynamic state*" of objects, as shape and position encodes their spacial "*static state*". In this category we deliberately leave apart the perception of local physical entities like touch pressure as they do not provide any global information about the environment and alone should be regarded as Blind Perception. The reason for this is the limited exploitability of such information, as already discussed in the appropriate section.

We must notice that this classification is partially hierarchical: a model exploiting physics perception must perceive also geometry, and in turn a model perceiving geometry also perceives appearance. The category of Blind Perception instead includes all systems that do not access global information and thus cannot be included in the others. When classifying a work we label it with the most restrictive category in which it can be included.

2.3.2 *Blind Perception*

Blind perception research works under the assumption of knowing only local contact or joint information, with no concern about the environment. Some works like Arimoto et al. [20, 21] deviate from the usual objective of statically stable closures and suggest a control theory approach on two-parallel-finger grippers, which cannot achieve force closure statically. Although interesting, the idea is limited by the need of grippers with fine and strong finger control and only two parallel fingers, which is impractical. As a result, the authors could validate their methods theoretically and by simulations, but no experiment on real robots has been done.

More recent works like Dang et al. [34, 37] pose the objective of evaluating or improving a given grasp in terms of its closure and robustness. In their first work [34] the authors provide a machine learning approach using Support Vector Machines to evaluate the robustness of a grasp based on tactile feedback. Data are collected through simulation, and the generalization of

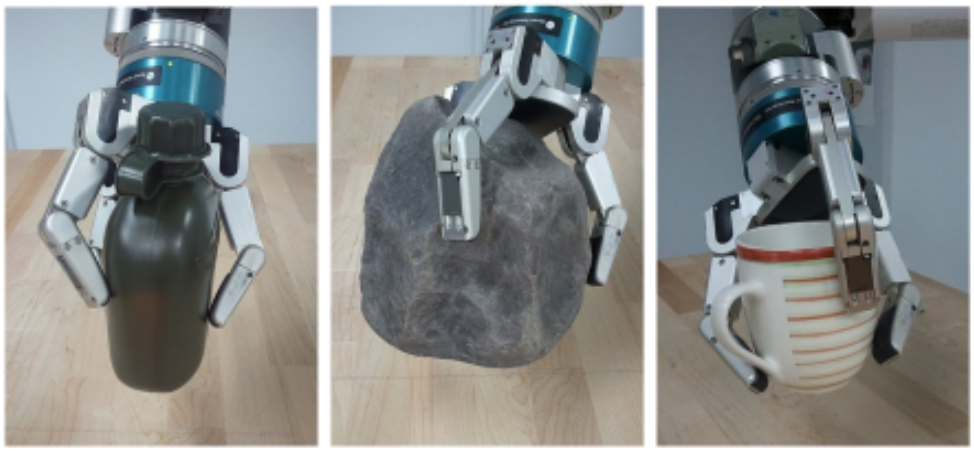


Figure 2.1: Examples of grasps obtained by iterative blind correction of random grasps by Dang et al.; image taken from [37]

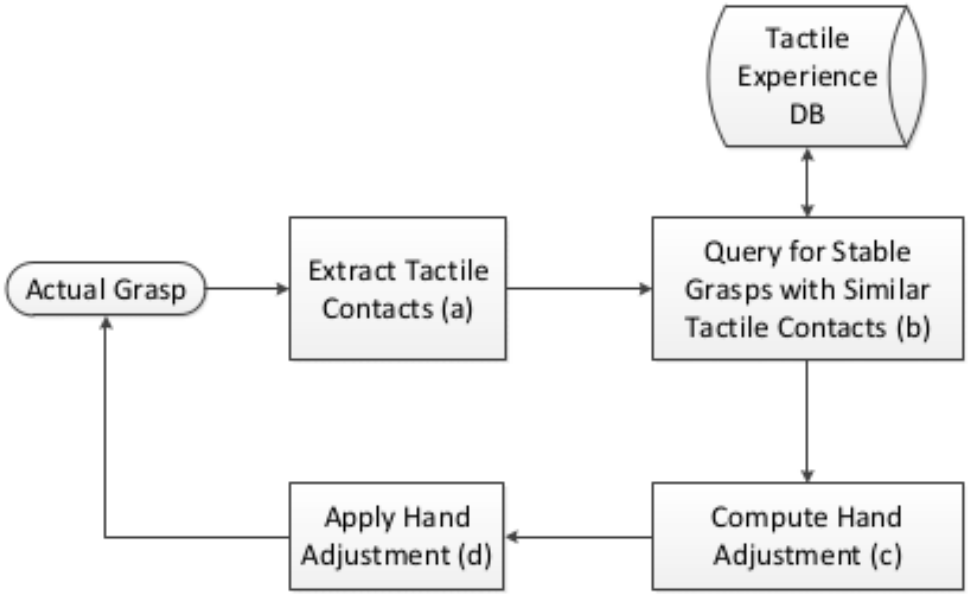


Figure 2.2: Diagram of the mechanism for iterative blind grasp correction by Dang et al.; image taken from [37]

their approach has been tested again only in a simulated environment with a model of the three-fingered Barrett Hand [10]. Their next work [37] (Figure 2.1), instead, subsumes the task of evaluation by trying to *improve* a given grasp. Dang uses the same pattern of collecting significant data from simulations and extracting operative knowledge from them, this time through the use of a K-nearest neighbors. By mimicking a hand pose and joint configuration similar to the known stable grasps which are nearest to the current grasp as shown in Figure 2.2, the hand is supposed to end to a more robust configuration. The authors proved, with experiments on a real Barrett hand, that random grasps on novel objects can be significantly improved in terms of robustness with their blind policy.

A common limitation of all these works, which is intrinsic to the category of blind perception, is the inability to plan any new grasps on objects whose position in space is not known a priori. A good opportunity, instead, comes from the reduced (though not insignificant) uncertainty of local measurements, which allows good generalization of simulation data to real applications.

2.3.3 *Appearance Perception*

The works under this category consider global knowledge of the world, but no explicit geometrical notion of distance and space is considered. This category received very limited attention from the research community and only few works are available as the great majority of researchers considering vision explicitly model and estimate at least some key spacial cues.

A relevant work in this context is the one by Levine et al. [61] who trained a deep learning based controller by leveraging on an extremely large scale data collection phase. The robots they used are arms with two-finger grippers positioned in front of a box with different small objects, with a single RGB uncalibrated camera facing towards the box as shown in Figure 2.3. The learned controller had the objective of successfully gripping every time a different object to lift it and place it back again. The authors proved that a learned controller could generalize well on different camera illumination and calibration conditions and different finger tension or tearing levels.

This work shows that under constrained settings it is possible to extract an implicit model of the required information for grasping from data even if the input information is extremely uncertain, variable, and incomplete. We must still take into consideration that this approach is limited by the extreme effort in collecting the required data and by the inability to produce a single model that generalizes over different tasks and settings.

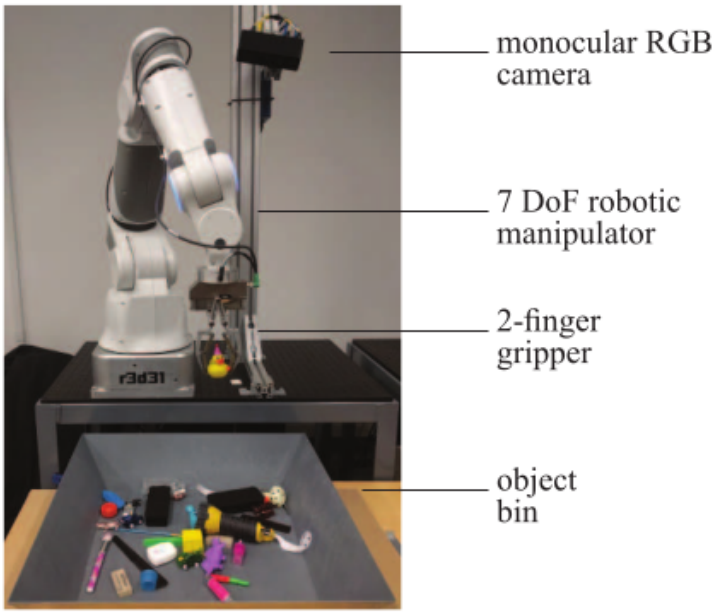


Figure 2.3: Experimental setting for the work of Levine et al.; image taken from [61]

2.3.4 *Geometry Perception*

Most of the works in Robotic Grasping belong to this class, thus very different approaches and settings have been defined. Some models extract just essential spacial information, like Kim et al. [30] who roughly estimate the 3D position of the target object from stereo vision for the first gross hand approaching movement. They use the interesting idea of positioning a stereo camera on the hand itself, to be able to detect the target in the environment and then focus on it while approaching, thus we can classify this work as being *mixed span*. Another important contribution is the one from the MIT Princeton team at the Amazon Challenge 2017 [62]: their work has analogies with the one from Levine [61] as it works in unstructured environments with many objects cluttered in a container, but they use four RGB-D cameras for a more informative perception with global span and directly infer gripper or suction affordance maps from deep models while employing simpler arm controllers to move as shown in Figure 2.4: in this case, space, differently from Levine, is explicitly known.

Other works in this category go in the direction of grasping complex objects after the analysis of their surface. Erkan et al. [31] propose to detect short segment edges on the surface of the object through Early Cognitive Vision descriptors and classify pairs of co-planar segments, shown in Figure 2.5, according to the quality of the grasp they afford, through semi-supervised learning. Their approach has a clearly focused span on a single

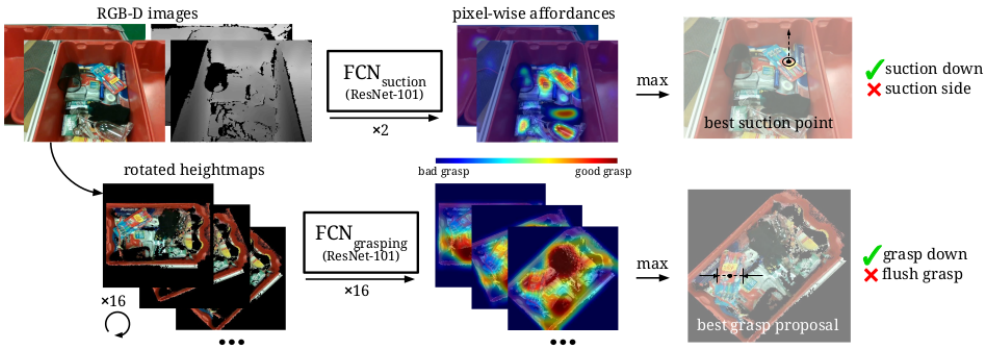


Figure 2.4: Grasp planning mechanism from the MIT Princeton team at the Amazon Challenge 2017, image taken from [62]

object whose interesting surface features are mapped in the space and jointly suggest grasp possibilities.

The works seen so far do successfully grasp objects, but they ignore completely that different grasps are required for different tasks: they fix the task of grabbing and eventually moving a target object, but they can hardly be generalized to different tasks. Biasing grasps towards the completion of some task is a key aspect for the relevance of grasping as discussed in Section 2.1. One of the first works integrating grasp planning with tasks is Prats et al. [25]. They use simple hardcoded 3D models of home objects (doors, drawers, windows) and preshapes of standard hand configurations to enact some task encoded in physical interactions like applying force or torque on specific degrees of freedom of the object, as shown in Figure 2.6. Their heuristic method has been successful on experiments with a real Barrett hand with a very rough model of the target object, but still it requires a 3D model of the object and it does not account for the high uncertainty of directly perceiving the model. We do not classify this work as Blind Perception as it takes into account geometrical knowledge of the environment, even if endowed and not perceived, nor it falls in the category of Physical Perception as physical quantities are used to model the task, not the object, although such physical quantities are the primary link between a task and its associated grasp passing through preshapes.

2.3.5 *Physics Perception*

The physical understanding of the environment is recently receiving attention from the Computer Vision community, trying to estimate various quantities such as mass [58], material [39, 44] or manipulation forces [47]. However, rarely researchers in the Robotic Grasping fields use similar techniques to physically model the environment.

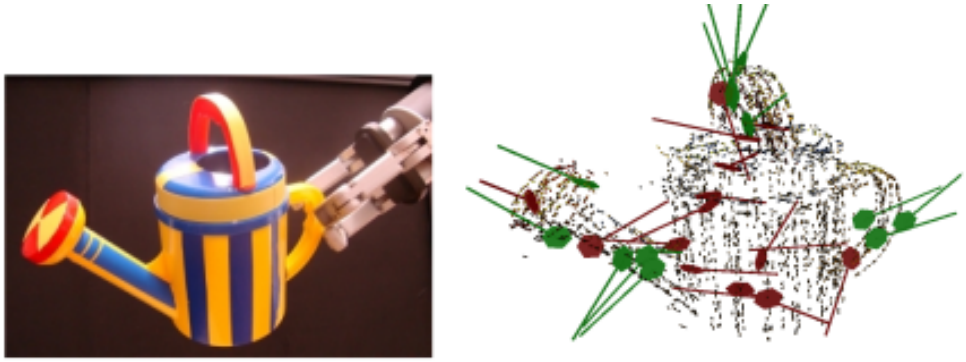


Figure 2.5: Example of positive and negative edges detected by Early Cognitive Vision features in the training set of Erkan et al.; image taken from [31]

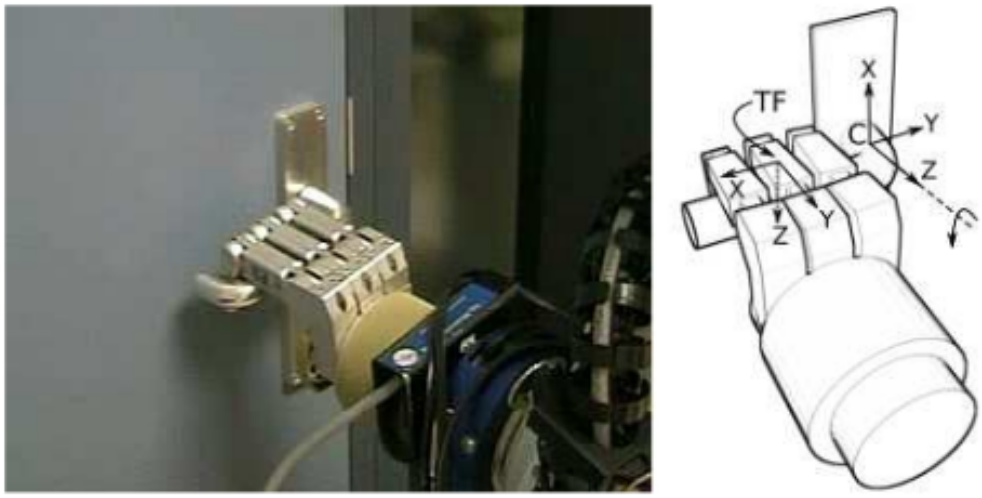


Figure 2.6: Example of physical representation of opening a door through the handle with associated preshape, image taken from [25]

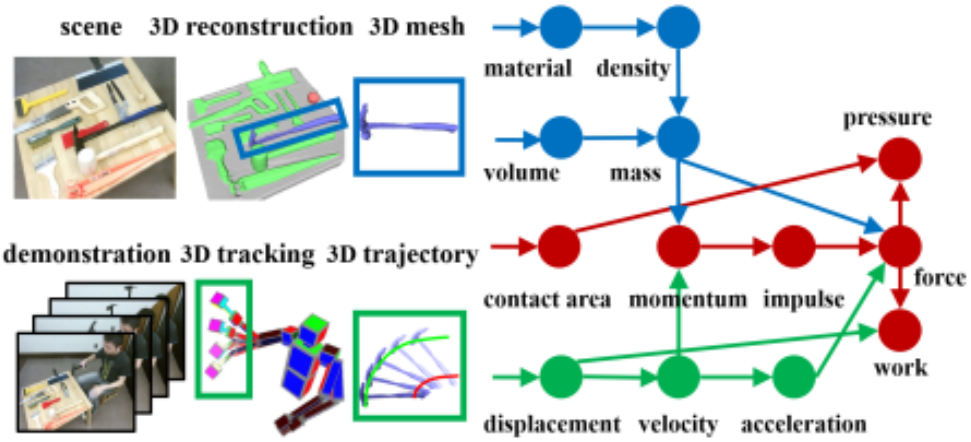


Figure 2.7: Physical quantities analysis pipeline in Zhu et al.; image taken from [50]

A significant work in our analysis is the one by Zhu et al. [50], in the more general framework of affordance learning, but still relevant for task-oriented grasping. They provide a model to estimate the best suited tool for a task among the ones presented on a planar surface through vision. The task is presented to the system through a video of a human demonstrator choosing the best tool among a different set and using it to perform a task like nut-cracking. The model involves the optimization of physical quantities in imagined tool uses as shown in Figure 2.7, and the choice of the best area of the tool for grasping and for functional use. The method is validated through benchmarking of the system choices compared against the ones taken by humans. The main limitation in this is the lack of an intrinsic definition of task which would enable further elaboration and adaptation to robotic actuators which are different from humans; moreover we miss a concrete connection between the planned grasp area on the tool and the actual grasp pose to effectively execute the task.

2.3.6 *Open Problems in Robotic Grasping*

As we have seen, the problems of evaluating the robustness of a grasp and planning grasps with perfectly known object models have been fully assessed, while consistent research efforts are currently producing good results towards the same problems under uncertain object models. On the contrary, the problem of task-oriented manipulation is still an open problem in the Robotic Grasping field: few works have been attempted, lacking a generalized framing of tasks either limiting task expressivity [50] or categorizing action possibilities [25]. Moreover, the modeling of physical quantities, which explain the connection between actions, tools and tasks, received almost no attention from the Robotic Grasping community and remains an unexplored opportunity.

2.4 RELATED WORKS IN AFFORDANCE LEARNING FOR GRASPING

As seen in Section 2.3.6, our broader research landscape lacks general frameworks for task-oriented manipulation and in particular the modeling of physical quantities is still a widely open path in this field. Moreover, as we are in a context of uncertain and incomplete perception from the environment, here we focus on a *learning* approach to infer useful knowledge from sensor inputs. Therefore, in this section we focus more specifically on the much more specific field of Affordance Learning in order to define more precisely the main limitations of the works most closely related to ours.

2.4.1 *Analysis of main related works*

Many researchers have worked towards the understanding and formalization of the concept of affordances [1–3, 5, 7, 12–17, 26]; they have been inspiring for roboticists to work within the affordances framework to define the autonomous interaction of a robot with an unknown environment.

Within this context, one of the first approaches towards task-oriented grasping, as we have already seen in Section 2.3, is reported in [25]; they proposed to encode the task in physical terms (e.g., applying a momentum on a handle to open a door, as in Figure 2.6) and then to solve the problem of grasp planning by hardcoding hand postures and their association with tasks; the method has shown good performance in the expected domain, but poor generalization capabilities.

Later works formalized the problem via graphical models, distinguishing task, object features, action features and constraint features. In particular, authors of [33] proposed the use of such formalization and they have been able to effectively learn to infer the likelihood of grasp approach directions with respect to a human-labelled ground truth as shown in Figure 2.8. The main limitation of this work, in our opinion, is the human intervention, which makes the real definition of the tasks implicit and prevents the scalability of the dataset that can be generated for learning without tedious human teaching. The direct intervention of human judgment on semantics to evaluate the quality of grasp hypotheses with respect to a given task is nevertheless a common approach to many research works, like [36] and [40] in which authors prove the effectiveness of a human-labelled semantic approach with real robot manipulations.

More recently, [54] has proposed to label mesh vertices in simulated objects as being graspable or not according to some task as shown in Figure 2.9, so that many scene examples can be produced and automatically labeled via simulation. This allowed the system to automatically segment graspable and not graspable regions of objects in cluttered scenes, but still the expressivity of this method is restricted to specifying graspable or not graspable surfaces.

Authors of [64] proposed a bottom-up approach for affordance perception by object parts which detects the local geometry of patches of the object and provides pixel-level affordance segmentation for pre-defined affordance classes. Their method is again based on a dataset [46] of 10000 pixel-labelled RGB-D images which have been hand labelled.

2.4.2 *Common limitations*

A common limitation of the works analysed in Section 2.4.1 is the vague definition of task affordances which passes through the human labeling of

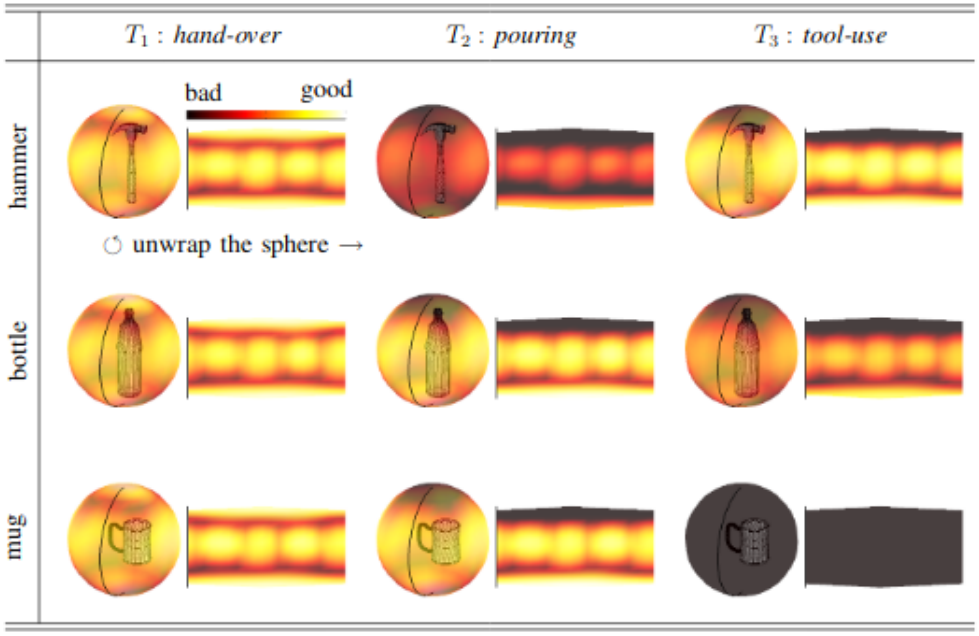


Figure 2.8: Affordance maps inferred by Song et al., referred to the suitability of a grasp direction towards some task on some object; image taken from [33]

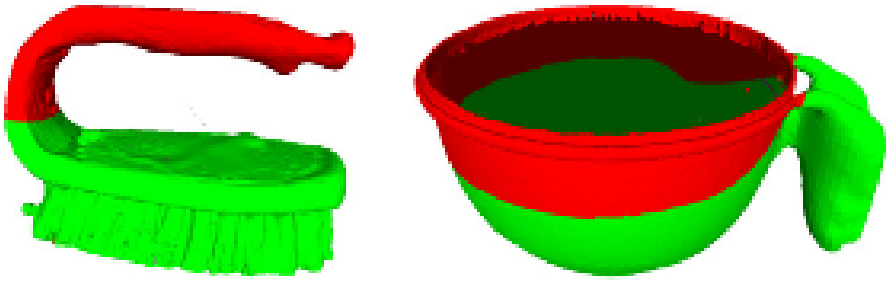


Figure 2.9: Labelled object meshes according to task constraints, on the left for the *handover* task, on the right for the *pouring* task; image taken from [54]

the ground truth. Defining task affordances through human-labelled ground truths intrinsically poses many limitations:

- **Optimality**: humans are used to categorizing objects and object usages, as well as grasps on standard objects. While on many standard tools which are built on purpose for some tasks we can expect that the human-chosen grasp is sufficiently optimal for the specific objective task of the tool, we largely lack this guarantee for non-standard objects or for non-standard object-task couples.

- **Bias**: even standard grasps on standard objects which have been designed on purpose for such grasps are biased by the embodiment of the human hand and arm. Human hands and arms largely differ from common robotic arms for the number of fingers, available degrees of freedom on both fingers and the arms, and extremely different friction and softness. Such differences completely change the statics of grasps and the dynamics of manipulation, changing task execution and, consequently, the available affordances. Such bias is not only present on the design of standard objects, but it is also intrinsic on the judgement of human labelers who are unavoidably biased towards their own manipulation experience.

- **Scalability**: a common limitation of any system requiring human labeling of data is scalability of the size of the produced dataset. Even producing a dataset of the order of the tens of thousands of examples is a very expensive and time-consuming task to require from humans. Machine-generated data, on the other side, have the potential to be produced at scale by many cores/servers in parallel and can easily surpass the size of human labelled datasets by many orders of magnitude.

2.5 CONCLUSIONS

As we have seen, the research field of Robotic Grasping is very broad and very active. We have identified Task Oriented Grasping as a still open field, and the use of physics-based models as an unexplored opportunity. Within Task Oriented Grasping we have focused on Affordance Learning, and we have seen that a common limitation of current research works is the use of human-labelled data.

As a consequence, the main objective of this thesis is to provide a novel framework for Task Oriented Grasping that could enable scalable and unbiased grasp affordance data collection for affordance learning, while reducing the human intervention in the definition of the tasks. Such automated data collection is based on state of the art physical grasp models that allow the

execution of grasps in a perfectly known simulated environment from which ad-hoc physical measures are extracted and logged as grasping data.

3

COMPUTATIONAL MODELS FOR IMAGES AND POINT CLOUDS

In this thesis we make use of models intended for image and point cloud processing, therefore here we provide a brief explanation of such models. The necessity of computational models for images and point clouds comes from the idea of enabling computational processing of the data coming from vision sensors, among which here we distinguish mainly RGB sensors (colored images cameras) and range sensors (depth camera). The main feature of such sensors is that they provide dense spacial information about the environment that can enable a much deeper understanding of it from the machine, in perfect analogy with vision for humans, at a relatively low price. However, much of the useful information that humans take from vision, like, e.g., the identification and localization of objects, are high level concepts that need to be extracted from raw vision, hence the need of the computational models to extract such knowledge from vision sensors. The aim of this chapter is not to make a full review of said models of Computer Vision, but rather to provide a more in-depth description of the main models employed in this thesis.

3.1 REPRESENTATION

Images bring dense information discretized in the form of a rectangular grid of pixels, where each pixel is itself a vector composed of multiple channels. Therefore, we can represent an image of size $w \times h$ and c channels as a tensor $I \in T_{w,h,c}$ where the first two coordinates are jointly significant and bring spacial information, while the last coordinate distinguishes between different channels of information corresponding to the same location. This distinction becomes important when speaking about Inception models in Section 3.2.3.

The most common use case for this representation is for RGB images, which encode separately the information about red, green and blue color into three separate channels, therefore having $c = 3$. Together with their equivalent representations (like HSV images which encode equivalently hue, saturation and value in the three separate channels) and reduced representations (like grayscale, which reduces a linear combination of color channels into a single value channel) color images represent the most common and important use case that first drove research in this direction and still takes

much research attention, however in this thesis we are mainly interested in range images, which represent more directly the geometry of the scene.

Range images only have one channel, encoding for each pixel location the distance from the camera to the first obstable encountered by the viewing ray in the scene up to a maximum sensible distance. These images describe the geometry of the environment more directly than color images, thus they are used in this thesis to provide partial information about the object geometry to neural models that are intended to process such information.

3.1.1 *Point cloud representation*

An even more explicit representation of the same information contained in a range image is a point cloud. Point clouds are sets of points, each point represented as the three-dimensional vector coordinates of its location in the real world with respect to some coordinate axes.

While point clouds are a different representation and concept than range images, they are strictly more powerful and any range image, with known camera parameters, can be mapped to its equivalent point cloud by mapping each pixel to its backprojection in space. If the camera is in the origin and looking along the z axis, (C_X, C_Y) are the pixel coordinates of the virtual camera center and f_X and f_Y its focal lengths, then each pixel (u, v) with depth channel value $d(u, v)$ is mapped to a point (X, Y, Z) as follows:

$$
\begin{aligned}
Z &= d(u, v) \\
X &= \frac{(u - C_X)Z}{f_X} \\
Y &= \frac{(v - C_Y)Z}{f_Y}
\end{aligned}
$$

while the reverse is not possible in the case of a general point cloud.

Point clouds are usually used also to incorporate in a single coherent representation the information extracted from multiple images, as there is no constraint on the represented locations while images are constrained to a rectangular grid of points and exactly one point on each viewing ray, thus producing models that are able to process general point clouds would allow them to be extended naturally to the case of multiple views.

3.2 CONVOLUTIONAL NEURAL NETWORK

Convolutional Neural Networks (CNN) are neural networks specifically designed to capture local patterns, encoding them in subsequently higher level feature spaces. Although they have been initially designed specifically to

capture bidimensional patterns along the two spacial dimensions of images, the concept is valid for any number of dimensions, and there are examples of both one-dimensional convolutions to analyze patterns in sequences like in speech analysis and three-dimensional convolutions to find volumetric patterns in voxel grids. Here we focus only on the standard bidimensional convolutional neural network as these are the models we are going to use. In this section we do not go into details about neural network models in general, which are taken as a prerequisite.

3.2.1 *Convolutions*

Convolutions are a basic image processing operation in computer vision. Let $I_1 \in T_{w_1,h_1,c}$ and $I_2 \in T_{w_2,h_2,c}$ with $w_2 < w_1$ and $h_2 < h_1$, then the convolution $I_1 \circledast I_2 = I_3$ is defined as follows:

$$I_3^{i,j,0} = \sum_{k=0}^{w_1} \sum_{l=0}^{h_1} \sum_{m=0}^{c} I_1^{w_1-k,h_1-l,m} I_2^{k+i,l+j,m} \tag{3.1}$$

While this general definition of convolution between two images can be useful in some cases, we are mainly interested in using one image $I \in T_{w,h,c}$ as an input image and an image $\omega \in T_{2w_k+1,2h_k+1,c}$ (that we usually index with symmetric index ranges centered around zero) as a kernel that determines the kind of operation that we are performing on the input image I.

In this specific context we can reframe the convolution $\omega \circledast I = J$ as follows:

$$J^{i,j,0} = \sum_{k=-w_k}^{w_k} \sum_{l=-h_k}^{h_k} \sum_{m=0}^{c} \omega^{k,l,m} I^{i-k,j-l,m} \tag{3.2}$$

Convolutions, optionally combined with a non-linear function, are widely used as a basic operator for image manipulation and to extract features from images. Classically the kernel ω is hand-designed to extract some specific feature like image derivatives or to obtain some effect like gaussian blur; some grayscale examples are reported in Figure 3.1.

3.2.2 *Basic Convolutional Neural Network*

The main intuition of a Convolutional Neural Network (CNN), first proposed by Yann LeCun [6] with the LeNet architecture, is to let the network learn the convolutional kernels that can extract the image features which are useful to accomplish the learning task: the kernel of the convolution becomes the parameter tensor which is optimized to minimize the loss function.

(a) Cameraman, original **(b)** 5×5 gaussian blur **(c)** sharpen

(d) Sobel vertical derivative **(e)** Sobel gradient magnitude **(f)** Sobel horizontal derivative

Figure 3.1: Common convolution operations in image processing. Notice that the Sobel gradient magnitude (3.1e) is obtained as the magnitude of the vector of the vertical and horizontal Sobel derivatives.

Such approach has been demonstrated to be far superior to classical fully connected neural networks as far as GPU training and large enough datasets became available [38], thus it attracted much research attention in the following years.

A convolutional layer maps an input image feature space $I_{in} \in T_{w_{in},h_{in},c_{in}}$ into an output image feature space $I_{out} \in T_{w_{out},h_{out},c_{out}}$ through a kernel matrix $K \in T_{w_k,h_k,c_{in},c_{out}}$ by performing c_{out} basic convolutions:

$$I^{\cdot,\cdot,i}_{out} = K^{\cdot,\cdot,\cdot,i} \circledast I_{in}, \tag{3.3}$$

where $\circledast$ is the basic convolution operation as defined in Equation 3.2. The dimensions w_{out}, h_{out} depend on the kernel sizes, padding strategies, and strides (regular subsampling of the output space). A stack of such linear operation is completely equivalent to a linear MLP applied to a small patch of the input image, on every possible patch location with shared parameters. Sharing parameters to this level allows the number of parameters not to scale with the dimension of the image, while incorporating into the architecture the very local nature of the strong correlations that exist between pixels. This is a strong architectural prior that greatly fits the image domain, substantially reducing the dimensionality problem which is intrinsic in images.

The overall CNN is not only composed as a plain succession of convolutional layers, as they would just result into mapping one image to a different image feature space through a linear transformation. A plain CNN is usually composed by the following elements:

- convolutional layers: as described above, they constitute the main element of the network
- non-linear activations: each convolution is usually followed by a non-linear activation function to introduce non-linearities into the network; as the convolutions stack is usually very deep, the most common is ReLU due to the problems of shrinking and exploding gradient with non-unitary derivative activation functions
- pooling layers: these layers perform subsampling of a feature space over the spacial dimensions, usually reducing regular squares of the input space into their maximum or their average. Such layers greatly increase the area of the input image that activations after the layer depend on (receptive field). Notice that, although they are commonly used this way, there are relevant cases such as that of the Inception in which they are not used to subsample, but rather to provide alternative aggregated features.
- a MLP: when the network task is not compatible with having an image in output, the last layer cannot be a convolutional layer. In these cases,

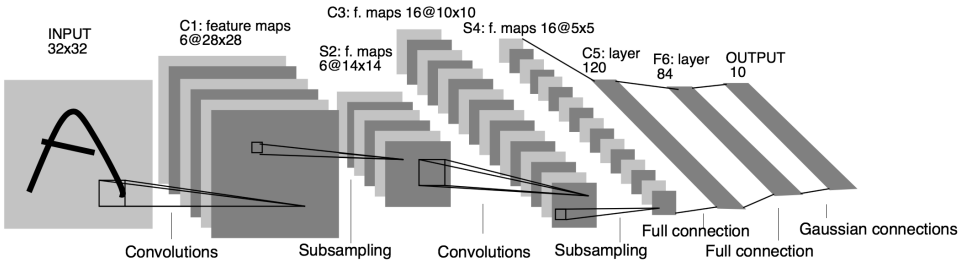


Figure 3.2: LeNet architecture, image taken from [6]

which are extremely common (e.g., image classification), the output image of the last convolutional layer, with much lower dimensionality and spacial correlations than the input, is used as feature vector for a MLP, which performs the task with meaningful features.

The structure of a plain CNN is summarized in Figure 3.2, which is the architecture of a LeNet [6], the first proposed convolutional neural network.

3.2.3 *The Inception Networks*

As anticipated before, many variations of CNN architectures have been proposed after the first successes with AlexNet [38], both in task and in structure. In this Section we review some structural improvements following the Inception networks that we use later in Chapter 6. The main objective of such modifications is to improve in the task of classification over the ImageNet dataset [29], which comprises millions of labelled color images over one thousand different classes.

3.2.3.1 *Inception-v1*

The first Inception architecture [49] was proposed for the ImageNet Large-Scale Visual Recognition Challenge 2014; it is composed of modules based on the following principles:

- Images from the same class in ImageNet usually display great variability in the size of the salient region. A solution inspired by a neuroscience model of the primate visual cortex is having many parallel paths with a different receptive field each
- Using larger filter sizes can produce a computational bottleneck when the number of filters raises significantly. Under the assumption that activations can be clustered and thus reduced in dimensionality, 1×1 convolutions can reduce the computational load and allow the training of deeper networks in a reasonable time

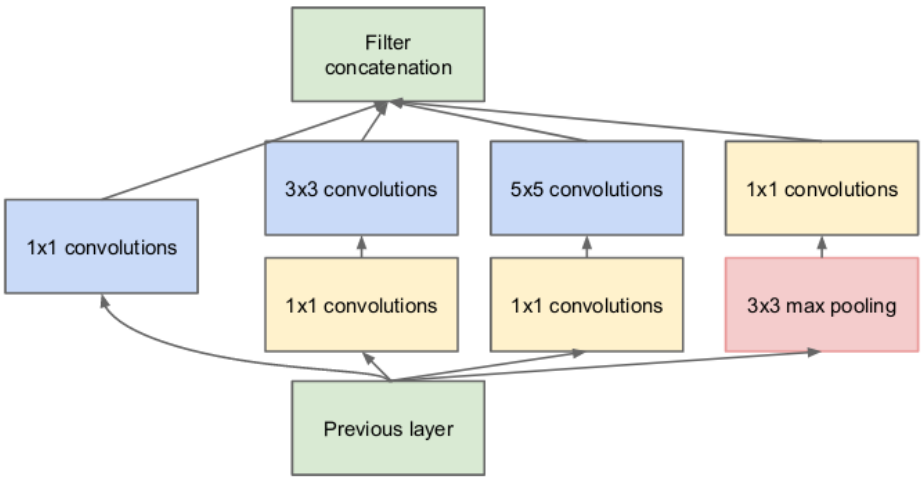


Figure 3.3: Inception v1 module, image taken from [49]

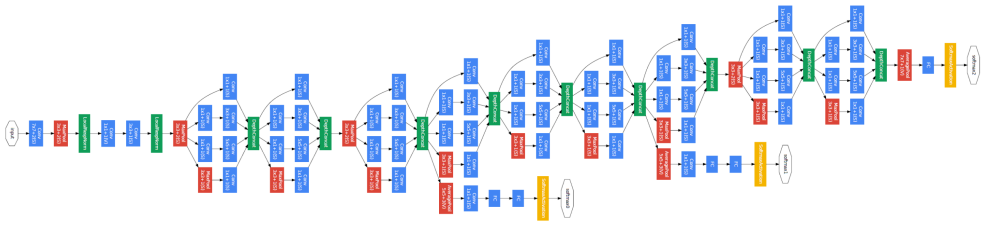


Figure 3.4: GoogLeNet architecture, image taken from [49]

The result is the Inception-v1 module shown in Figure 3.3. It contains three alternative convolutional paths with different filter sizes ($1 \times 1, 3 \times 3, 5 \times 5$), where the largest two are anticipated by a 1×1 convolution for dimensionality reduction. The authors [49] stack such modules obtaining a 22-layers deep CNN that they name GoogLeNet (Figure 3.4) which is trained with the help of auxiliary classifiers to address the problem of the shrinking gradient.

3.2.3.2 *Inception-v2 and Inception-v3*

Inception-v2 and Inception-v3 were proposed in the same paper [52] to incorporate a number of improvements based on the following principles:

- Avoid representational bottlenecks for any cut in the model, especially early in the network. This is done by *gently* reducing the dimensionality of the representation towards the output, even though this rough criterion ignores other factors like the correlations between components

- Higher dimensional representations are easier to process locally in a network, allowing to disentangle relevant features and speeding up training
- Spacial aggregation can be perfomed over lower dimensional features with not much loss in representational power as they can leverage the strong spacial correlations between units
- Depth and width of the network should be balanced to improve the representational power without introducing representational bottlenecks, and vice-versa

Such principles drove the design of new Inception modules used within the Inception-v2 architectures, where Inception-v3 is a model with a tuned combination of all the improvements they introduced.

The three new modules are shown in Figure 3.5. The module in Figure 3.5a is obtained by the original Inception-v1 module of Figure 3.3 and decomposing the 5×5 convolution into two successive 3×3 convolutions, using less parameters overall, and with lower dimensionality in between to reduce the feature vector size at the presence of spacial convolutions; it is used in the early stages of the network. The module in Figure 3.5c is used just after the one described above, and applies the same principle even further by decomposing very large $n \times n$ convolutions into $1 \times n$ followed by $n \times 1$, in particular the Inception-v3 network uses such modules with $n = 7$. The module in Figure 3.5b is used just before the final MLP classifier and enlarges the dimensionality of the available features to disentangle the correlations for the final classification.

Moreover, Inception-v3 includes some minor improvements such as batch normalization within auxiliary classifiers, label smoothing and a different optimizer (from momentum to RMSProp).

3.2.3.3 *ResNet*

The ResNet architecture, proposed in [51], is a separate branch from the Inception series that introduce the concept of residual learning and inspired the design of the following Inception.

The main concept behind residual learning is that deeper networks are more difficult to optimize due to the stack of composed function that they represent, independently of the problem of the shrinking or exploding gradient. Therefore, the results of training deeper networks get worse (degradation) even in terms of training set performances, with the present optimizers. This is in contrast with the theoretical reasoning that a network $\mathcal{N}'$ which is one layer deeper than a network $\mathcal{N}$ should be able to perform at least as well as $\mathcal{N}$ in training error, as it could just learn the identity function in its extra layer and reduce to the case of network $\mathcal{N}$.

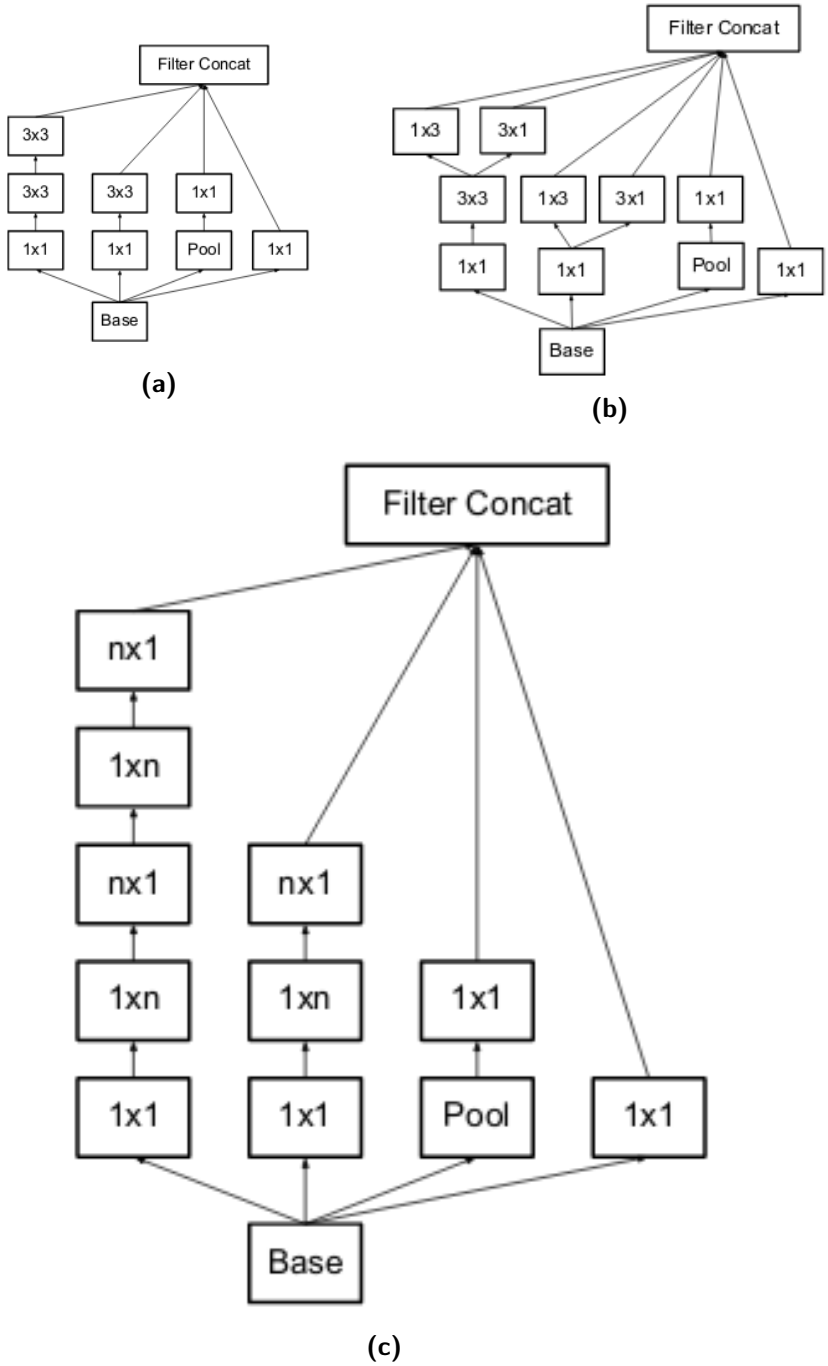


Figure 3.5: Inception v2 modules, image taken from [52]

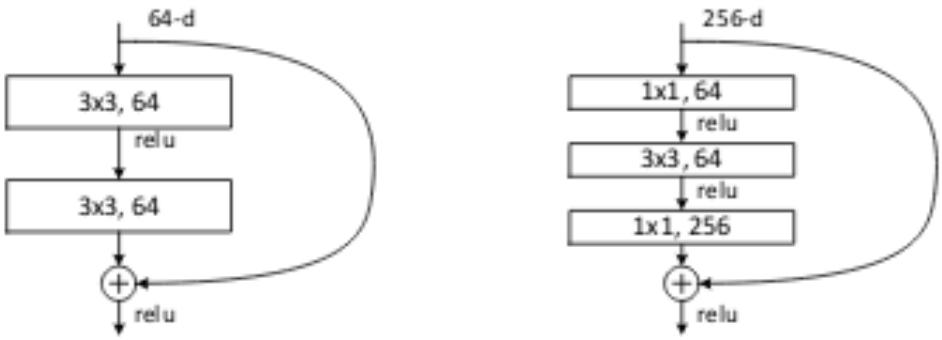


Figure 3.6: ResNet modules, with (right) and without (left) dimensionality reduction inspired by Inception modules, image taken from [51]

The solution proposed by the ResNet authors is to help the optimizer to find this backup solution in complete analogy with the theoretical proof by decomposing each layer that has to learn a function $\mathcal{H}$ over an input x into the identity and a residual function $\mathcal{F} = \mathcal{H} - x$ so that the backup solution of learning the identity is equivalent to shrinking the learned function $\mathcal{F}$ to zero.

This results into the design of the ResNet modules shown in Figure 3.6, which use an identity skip connection to leave to the convolutions the task of learning the residual function $\mathcal{F}$; this led to the successful optimization of much deeper architectures with over 100 layers, with some experimentation over 1000 layers.

3.2.3.4 *Inception-v4 and Inception-ResNet*

The Inception-v4 and Inception-ResNet [59] architectures were presented together in the same work. Inception-v4 is meant to apply the guiding principles of the previous inceptions to the stem of the network, which is the first set of convolutions, making the network structure more uniform. The main novelty is in the Inception-ResNet idea, that applies the approach from [51] explained in Section 3.2.3.3 to redesign the Inception modules as residual functions.

The new modules are shown in Figure 3.7: they generally reduce the complexity of the single module to stack more modules while having a comparable complexity, and substitute the pooling path with the identity skip connection. The resulting architecture performs comparably with Inception-v4 and has comparable complexity, but has much faster training time and can achieve better performances under the same training resources budget.

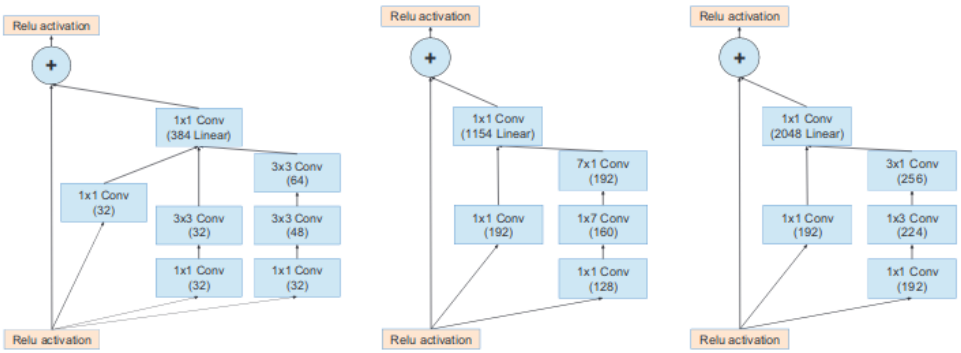


Figure 3.7: Inception-ResNet-v2 modules, image taken from [59]. Inception-ResNet-v1 modules are perfectly equivalent, with reduced number of filters.

3.2.3.5 *Extreme Inception*

The Extreme Inception [53] (Xception) architecture is largely inspired from the Inception and ResNet works.

The author reframes the Inception modules as following a weak assumption of independence between cross-channel correlations and spacial correlations, thus taking advantage from a weak separation of the processing of such correlations. The Xception architecture aims at bringing this assumption to the extreme, processing cross-channel correlations and spacial correlations in two distinct independent steps respectively of pointwise convolution and depthwise convolution.

The Xception basic building block is built starting by a simplified version of the Inception module (Figure 3.8a) which is transformed into an equivalent form where the outputs of each pointwise convolution is concatenated (Figure 3.8b) to highlight the weak assumption on the independence of cross-channel and spacial correlations which are separated in blocks, and finally bringing such scheme to the extreme of complete separation (Figure 3.8c).

The actual implementation of the Xception is by using standard depthwise separable convolutions, that is a succession of a depthwise convolution followed by a pointwise convolution (inverse order with respect to the extreme Inception module). Depthwise separable convolutions are regularly supported by skip connections that imitate the ResNet residual identity connections, with the difference of being 1×1 strided convolutions to match the output spacial and channel depth dimensions. The complete Xception architecture, that we use in this thesis, is reported in Figure 3.9.

3.3 POINTNET

The PointNet model [56] shown in Figure 3.10 has been designed specifically to effectively process unordered sets of points and tested on many tasks in-

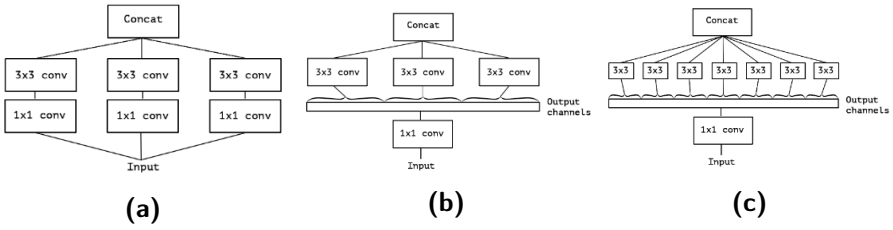


Figure 3.8: Strengthening the independence assumption on Inception modules, image taken from [53]

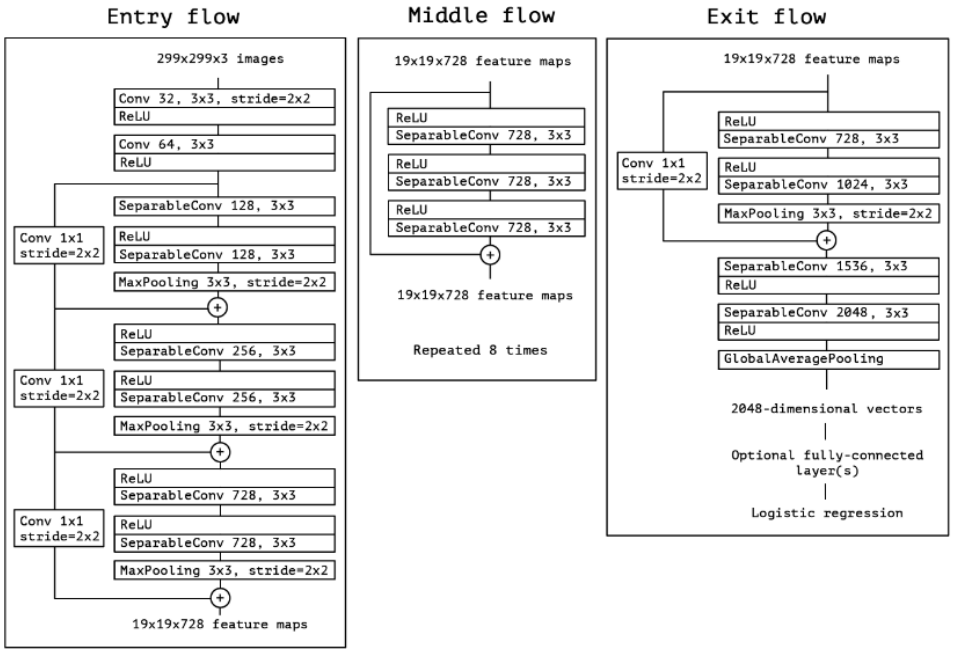


Figure 3.9: Xception architecture, image taken from [53]. All SeparableConvolution layers have depth multiplier of 1 and are followed by batch normalization [45].

volving inputs in the form of point clouds, such as point cloud classification, part segmentation or semantic segmentation.

More specifically, it is designed for classification and segmentation upon unordered sets of points in a Euclidean space in $\mathbb{R}^n$, thus satisfying the following properties:

- **Invariance under permutations**: as input data are unordered sets of points, the network needs to be invariant to different permutations in which the points may be presented as input.
- **Interaction among points**: the input data is expected to represent a complex geometry out of which subsets of points may display meaningful structures. Therefore, being able to capture the interactions among points is vital for the designed network.
- **Invariance under transformations**: as the classes of objects and single points (for segmentation tasks) do not depend on the reference system of the point cloud, the designed network also needs to be invariant to transformations of the system of reference.

If $x_1 \dots x_k$ are the input points in $\mathbb{R}^n$, then the network computes a function $f(x_1 \dots x_k)$ decomposed as:

$$f(x_1 \dots x_k) = g(h(x_1) \dots h(x_k)), \tag{3.4}$$

where g is a symmetric function and h is a shared feature extractor, thus granting invariance under permutations. In the PointNet architecture g is a max pooling between the extracted features of each point computed independently, while h is a shared MLP with some extra learned matrix multiplications which depend on global features (the outputs of T-Nets in Figure 3.10). Such matrices are applied on the coordinates of points and on later features to enforce invariance under transformations: a basic version of the network (MLP and max pooling) is used to compute some global features to feed to a MLP that outputs the transformation matrix, aware of the global context, that is expected to learn to normalize the reference system. Finally, global features are either used for direct classification or fed back to point features for the segmentation task, modeling interaction among points through the correlations between point features and global features.

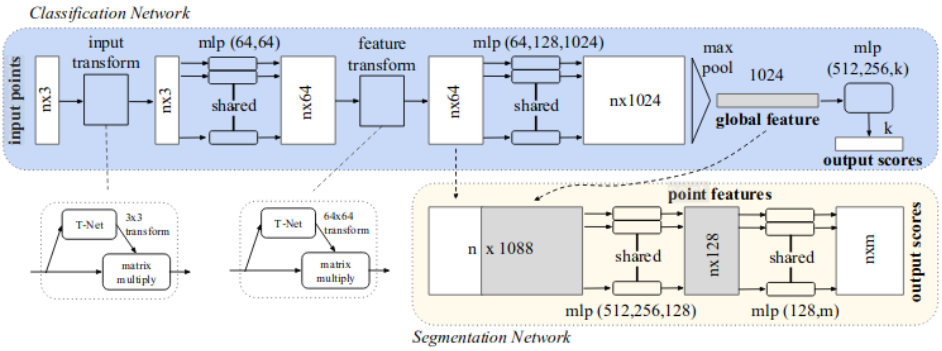


Figure 3.10: PointNet reference architecture. Image taken from [56]

4

PROPOSED APPROACH

As we have seen in Chapter 2, the research community has spent much effort in tackling the problem of grasping novel objects in different settings [30, 31, 37, 61, 62] with the aim of holding objects robustly with robotic manipulators; however, real manipulation tasks go far beyond holding the objects and the quality of a grasp depends on the task it is meant to support. While many quality metrics exist to evaluate the quality of a grasp by itself [9, 48], no clear quantification of the quality of a grasp relatively to a task has been defined. In this chapter we formulate our theoretical framework, extending the concept of quality metric to task-oriented grasping by defining general physical measures for an open set of task affordances. This lays the foundations of our method in a very broad and general sense in the perfect information context. Then we extend our own framework to a vision context of partial information to bring our theoretical tools nearer to real applications, defining the models that we use for inference. Finally, we specialize our framework onto the analysis of specific physical quantities oriented to some selected target tasks within the sample domain of the kitchen robot, as described in Chapter 1.

4.1 TASK ORIENTED GRASPING FRAMEWORK

Let $\mathcal{O}$ be the set of possible object surfaces with friction and softness properties defined at each point, let $\mathcal{G}(O)$ defined on an object $O \in \mathcal{O}$ be the set of possible grasps determined by the hand embodiment, degrees of freedom, contact locations on the object and contact nature (e.g., frictionless, hard contact or soft contact), and $\mathcal{U}(O)$ be the set of points on the surface of the same object that can be considered as locations where the object can be used to perform some task.

Let $O \in \mathcal{O}$, $G \in \mathcal{G}(O)$, $U \in \mathcal{U}(O)$, then we define the *affordance function* of task T as $F_T(O, G, U) \mapsto \mathbb{R}$ such that $F_T(O_1, G_1, U_1) > F_T(O_2, G_2, U_2)$ if and only if the grasp and use hypothesis (O_1, G_1, U_1) is more suitable than the hypothesis (O_2, G_2, U_2) to perform task T, thus defining an affordance ordering of an object grasp for task T. The objective of the affordance function definition is giving a score to any grasp and use hypothesis on different objects to be able to extract a best hypothesis by optimization over G and U upon F_T:

$$G^*, U^* = \underset{(G,U)\in\mathcal{G}(O)\times\mathcal{U}(O)}{\arg\max} \quad F_T(O, G, U) \tag{4.1}$$

As we want a compact representation of the affordance function that we can practically express, we approximate F_T as a $\tilde{F}_T : \phi \in \mathbb{R}^n \mapsto \mathbb{R}$ by mapping the triplet (O, G, U) into a metric vector $\phi \in \mathbb{R}^n$ through a function $\Phi(O, G, U) \mapsto \mathbb{R}^n$. This metric vector is a collection of metrics encoding the geometrical and static physical properties of the triplet (O, G, U) which are relevant to approximate F_T. Therefore, the vector ϕ is used as a feature vector describing the triplet (O, G, U) to compute the affordance represented by F_T (or, more precisely, its approximation $\tilde{F}_T$), while it is also a vector of metrics that we require to be perfectly interpretable, precisely defined to be potentially measured or explicitly computed from the triplet (O, G, U) in a perfect information context. The reason for such extra requirement on metrics ϕ is to enable the explicit computation of such metrics in a simulated perfect information environment and collect a dataset of such measurements, to subsequently learn from the collected dataset to infer metrics ϕ from vision; finally computing $\tilde{F}_T(\phi)$ with a hardcoded $\tilde{F}_T$ for any task T.

In Section 4.3.1 we provide some examples of basic metrics ϕ and examples on how they could be used to hardcode $\tilde{F}_T$ for some reference tasks.

4.1.1 *Achieving Object Semantics Independence*

The complete object geometry is generally not available in real world applications, in particular when our long term goal is to infer object affordance from vision with no hardwired semantics. To achieve such goal, we need to frame the problem in the context of uncertain and incomplete information about the object by decoupling the grasp and use location description from the exact object geometry and possibly its semantics.

4.1.1.1 *Decoupling grasps*

Recall here that the complete description of a grasp requires the geometry and nature of contact points on the grasped object, and the grasp itself needs to be actuated by a grasping policy. If we assume the grasping policy to be deterministic, then we can define it as a function $GP(p_0, O) \mapsto \mathcal{G}(O)$ that maps an initial state $p_0 \in \mathcal{P}_0$ and an object O into the final grasp $G \in \mathcal{G}(O)$. To decouple from the specific grasp, and its parameters, we fix a grasping policy that allows a sufficient exploration of the grasps space $\mathcal{G}$ via the space $\mathcal{P}_0$ of possible initial states, which we call *pregrasps*.

In particular, we select a simple, but effective, grasping policy depicted in Figure 4.1 defined as follows: from an initial position of the hand with open fingers (4.1a), we advance towards a fixed direction until the first contact is

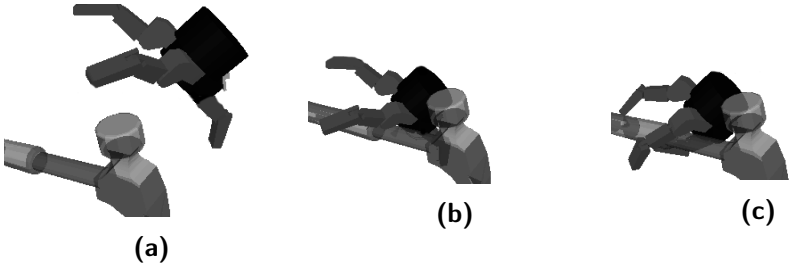


Figure 4.1: The three phases of our grasp policy.

made (4.1b), then the fingers are closed until all of them either make contact or are completely closed (4.1c).

4.1.1.2 *Extension to use locations*

Use locations are points on the bidimensional surface of each specific object; we decouple them from the specific object by defining use directions and a direction mapping function $DM(d, O) \mapsto \mathcal{U}(O)$ in complete analogy to the grasp decoupling solution. Our direction mapping assumes $d \in \mathbb{R}^2$ to be the spherical coordinates of a directed ray centered in the center of mass of object O and outputs the farthest point $U \in \mathcal{U}(O)$ which is the intersection of such ray with the outer object surface.

While grasps require decoupling from the object, as otherwise they would need to be expressed as a set of contacts precisely located on a surface that we do not know exactly, the localization of use locations do not require the same precision. As a consequence, although formalizing a decoupling method for use locations, we must take into consideration the possibility to express use locations directly as points loosely localized on the surface of the object. Such direct expression has the obvious advantage of being much simpler with respect to the one passing through the direction mapping function, which translates to less complexity in the learning models that we need to define for inference, thus we consider and compare both alternatives in two distinct settings that we describe more in detail in Section 5.3.1.2.

4.2 INFERENCE FROM VISION

As our goal is to make robots able to use unknown objects in a task consistent way, we need the robot to be able to perceive their affordances via sensors. In particular, we focus on vision being it an extremely common and effective tool to take information from the environment in real applications. We define our inference setting by focusing on the specific case of single range images taken from camera-in-hand perspective: such case provides only local geometry information about the object around the expected location of

the planned grasp. We want to learn a model that predicts the elementary metrics $\phi \in \mathbb{R}^n$ of a triplet (O, G, U) with only partial information about the observed object O. Indeed, predicting the vector ϕ would allow to estimate the affordance function F_T for any task T for which we can define an $\tilde{F}_T$.

Let $D : \mathcal{P}_0 \times \mathcal{O} \mapsto \mathbb{R}^{h \times w}$ be the function that maps a pregrasp p_0 on an object O to the depthmap of size $h \times w$ from the camera-in-hand perspective of p_0. Then we want to learn a model $\mathcal{M}^\Phi$ that approximates the mapping from $(p_0, D(p_0, O), d)$ to $\Phi(O, GP(p_0, O), DM(d, O))$ where O is an object, p_0 is a pregrasp, d is a use direction, and Φ is the metric extraction function, under the grasping policy GP and the direction mapping function DM.

We assume to be able to learn model $\mathcal{M}^\Phi$ from a dataset of tuples $< O, p_0, d, \Phi(O, GP(p_0, O), DM(d, O) >$ obtained via uniform sampling of p_0 and d values on a set of available objects models and computing the true values of $\Phi(O, GP(p_0, O), DM(d, O))$ via simulation. Details on our data collection setup are explained in Chapter 5.

4.2.1 *Model definition*

To structure the learning task, we define the model $\mathcal{M}^\Phi$ as the composition of two models: an input value $(p_0, D(p_0, O), d)$ is first classified by a binary classifier $\mathcal{M}^\Phi_C$ that outputs the probability for the input grasp of being a "good" grasp worth further evaluation or not. We define "good" grasps those respecting a minimum quality independently from the task, thus employing state of the art grasp quality metrics to generate the ground truth. The samples classified as positive then pass through a regression model $\mathcal{M}^\Phi_R$ that infers the metrics ϕ with the implicit assumption that the grasp is indeed a quality grasp.

The model $\mathcal{M}^\Phi$ is therefore constituted as in Algorithm 1, where g is an input grasp, u is an input use location, τ_C is a classification threshold to allow grasps from the classifier $\mathcal{M}^\Phi_C$ and v_{fail} is a special return value that indicates that the grasp is not suitable for any task.

Algorithm 1

```
function M^Φ(g, u)
    if M^Φ_C(g) < τ_C then
        return v_fail
    else
        return M^Φ_R(g, u)
    end if
end function
```

For both models $\mathcal{M}^\Phi_C$ and $\mathcal{M}^\Phi_R$ we propose and evaluate the use of many variations of the Convolutional Neural Network (CNN) and the PointNet

(PN) [56] architectures which have been discussed in Chapter 3. Both architectures encode the available geometry information (in the form of range image for the CNN or as the equivalent projected point cloud for the PointNet) in a feature vector, we condition the models with the other input parameters p_0 and d either early with the geometry input or late with the extracted features, and output the classification label or the inferred regression value with a classical Multi-Layer Perceptron (MLP).

4.3 TASK ORIENTED GRASP METRICS

After the definition of our theoretical framework for Task Oriented Grasping, we need to instantiate it concretely: we choose sample tasks from our application domain on which the system is evaluated, we select important geometrical and static features to incorporate into ϕ, and finally we define affordance function approximations $\tilde{F}_T$ for all chosen tasks T.

From our application domain of the kitchen assistant robot we want to select few very diverse tasks to test on. The typical pick-and-place, which we call picking, must be included in our tests as it is an extremely common task in any application domain. An effective pick does not only require grasp robustness, but also the minimization of the torque of gravity on the grasp location, which translates to grasping near the center of mass of the object. Cutting, on the other side, requires understanding the local geometry of the use location which should be sharp, and at the same time requires a grasp which is able to statically load as much pressure as possible on the cut location. Finally, the beating task requires a similar optimization, but directed towards the momentum that the beating motion would be able to discharge on the beating location, involving completely different dynamics.

4.3.1 *Basic Grasp Metrics*

We consider the following set of elementary metrics of (O, G, U), which have been selected based on the chosen tasks, and therefore should not be considered as exhaustive:

GRASP ROBUSTNESS ($\epsilon \in \mathbb{R}$) ϵ is a real number describing the robustness of the grasp. It is defined as the Epsilon metric described in [9] which is builtin in the GraspIt! simulator [18]. Force closure grasps have $\epsilon > 0$, where a higher value of ϵ imply that a greater minimum perturbance is needed to break the grasp.

ROTATIONAL INERTIA ($I \in \mathbb{R}$) I quantifies the rotational inertia around the axis of rotation of the wrist of the hand assuming a unitary density of

the object and assuming the hand to be integral with the whole object. It does not take into account the mass of the hand itself.

HAND EFFORT ON IMPACT ($E_i \in \mathbb{R}$) E_i describes the effort of the hand to balance the impact forces after a rotation around the wrist. It assumes a fixed average inertial torque in a small Δt during the impact which is directly proportional to I and a free contact force on the use location towards the normal direction. This metric takes the value of the minimum sum of the contact forces of the hand constrained to the contact friction cones to balance the inertial torque, ∞ if the minimization problem is unfeasible.

HAND EFFORT ON HOLD ($E_h \in \mathbb{R}^6$) E_h is a vector of six independent values which quantify the hand effort to balance a different gravity vector for each component of E_h. The hand effort is the minimum sum of all contact forces constrained to the contact friction cones that balance a given unitary force of gravity, ∞ if such problem is unfeasible. The six gravity vectors chosen are aligned with the three coordinate axes (once in the same direction, once opposite) of the object mesh as all meshes that we used in the Princeton Shape Benchmark have been designed by humans that gave a semantic meaning to the coordinate axes directions, aligning them with the sides of objects with an implicit notion of up, down, left, right, front and back. We use such design bias as a prior of where the gravity is more likely to be aligned, as objects are usually lying on one of such sides.

MOMENTUM DISCHARGE EFFICIENCY ($\delta \in \mathbb{R}$) δ quantifies the efficiency of discharging the rotational inertia of the wrist on the object use location. It quantifies the alignment between the inertial torque and the torque generated by a force aligned with the use location normal vector towards the inside of the object surface. It is computed as the dot product of the two normalized vectors, clipped to zero in case of negative values.

FORCE TRANSMITTED TO USE ($U_\tau \in \mathbb{R}$) U_τ quantifies the force that can be transmitted to the use location using constrained contact forces. It assumes all contact forces are constrained by their friction cones and have unitary maximum normal forces. It takes the value of the maximum force on the use location towards the use location normal guaranteeing static conditions.

USE LOCAL GEOMETRY ($U_g \in \mathbb{R}$) U_g describes how much the use location has the shape of an edge. It is obtained by fitting a quadratic function on the vertices of the triangles near the use location (including all the triangles that share at least one vertex with the triangle where the use location lies) and extracting the eigenvalues of the hessian matrix of such quadratic

function. The two eigenvalues λ_1 and λ_2 are the two principal component curvatures, so we quantify an edge with the expression $(\lambda_1 - \lambda_2)^2$ to identify locations with a great difference in local curvatures.

4.3.2 *Affordance Functions from Basic Metrics*

On top of the metrics defined in Section 4.3.1 we define the affordance functions for $T \in \{\mathtt{beat}, \mathtt{cut}, \mathtt{pick}\}$. In this preliminary study affordance functions have been designed by hand, to validate the feasibility of the framework, in future works we aim to learn them by optimizing task execution efficacy.

BEATING The classical beating action of a hammer with a human hand requires dexterous movements of the wrist which would need moving the whole robotic arm to be reproduced on any robotic hand with a reasonably simple wrist. For this reason we assume that the beating action is executed by the robotic actuator by simply rotating the hand clockwise around the wrist. We require that the hold is stable over a minimum threshold and that the rotational energy gets discharged almost entirely on the point of use, therefore we assign a fitness of $-\infty$ if the ϵ or δ are below some thresholds τ_ϵ and τ_δ. We want to maximize the ratio of the energy that we can incorporate into the rotation (assuming a maximum rotational speed it translates to the metric I) over the actual hand effort of keeping the object stable on the impact, which is the metric E_i. This is detailed in Algorithm 2.

Algorithm 2

```
function F̃_beat(ε, δ, I, E_i)
    if (ε < τ_ε || δ < τ_δ) then
        return −∞
    else
        return I / E_i
    end if
end function
```

CUTTING The action of cutting is extremely complex by itself and varies greatly with different materials and their surface and micro-structural properties. A complete physical study of this particular task is not our objective; we simplify it considering as approximation that greater force provides better cuts if executed on a thin enough edge. Therefore, we require a minimum robustness and a minimum edge score, assigning a value of $-\infty$ if metrics ϵ or U_g are below thresholds τ_ϵ and $\tau_{use\mathrm{Geom}}$ respectively. In case this condition is respected, the fitness for cutting is given directly by the U_τ metric

quantifying the force that can be executed by the grasp upon the use location. This is detailed in Algorithm 3.

Algorithm 3

1: **function** $\tilde{F}_{cut}(\epsilon, U_\tau, U_g)$
2: **if** $(\epsilon < \tau_\epsilon \ || \ U_g < \tau_{U_g})$ **then**
3: **return** $-\infty$
4: **else**
5: **return** U_τ
6: **end if**
7: **end function**

PICKING Picking an object (as the first part of the pick-and-place task) only strictly requires a stable grasp for a successful pick. However, different stable grasps may imply very different effort from the hand actuator to balance the force of gravity on the object. For this reason we require a stable grasp and minimize the sum of the contact forces required to balance the force of gravity in the six directions evaluated by the E_h metric. Notice that an unstable grasp needs to have at least one evaluated direction of gravity that the grasp cannot hold, thus we do not check the ϵ metric, while the fitness value for minimizing the sum of the hand effort on holding the object is given by the negative of the sum of E_h. This is detailed in Algorithm 4.

Algorithm 4

1: **function** $\tilde{F}_{pick}(E_h)$
2: **return** $-\sum_{i=1}^{6} E_h[i]$
3: **end function**

4.4 CONCLUSIONS

The theoretical framework described in this chapter has the aim of overcoming the limitations on the current state of the art highlighted in Section 2.5. The affordance of an object O with respect to a task T is expressed implicitly within the ordering that the affordance function F_T defines upon the triplets (O, G, U) with a fixed object O. Therefore, by hardcoding a function $\tilde{F}_T$ which acts as a feature-based approximation of F_T, we are effectively defining the affordance of a task T on objects. As discussed in Section 4.1 we require the features ϕ to be both interpretable and measurable or explicitly computable, thus we call them metrics, for two important reasons:

- Interpretability of metrics allows us to hardcode affordance functions $\tilde{F}_T$ for sample tasks like picking, cutting and beating, based on the intuitive human understanding of the meaning of such tasks. This reduces

the human intervention to the bare minimum of translating the fuzzy semantics of the idea of each task into an exact definition that can be used to automate the evaluation of such tasks from metrics. We discuss going beyond this interpretability requirement in Section 7.1.

- Explicit computability of metrics allows us to actually compute them in a perfect information simulated environment. This is a fundamental step to allow the inference of the same metrics in more realistic partial information settings such as from vision, as it enables the automatic collection of a dataset of computed metrics associated to pregrasps and use location hypotheses, used to learn models for the inference of computed metrics from simulated vision.

By hardcoding function $\tilde{F}_T$ and predicting the metrics ϕ from vision, we are finally able to estimate the fitness of a grasp and use hypothesis on an object with respect to a task by looking at the object, therefore we can obtain a best grasp and use candidate by estimating the fitness of a set of grasp and use hypotheses.

5

LEARNING FRAMEWORK AND DATA GENERATION

In this chapter we describe the process of data collection by building our implementation of the system proposed in Chapter 4 as a GraspIt! plugin for data generation. Moreover, we detail the learning framework around all the models that we experimented with in Chapter 6.

5.1 THE GRASPIT! SIMULATOR

As anticipated in Section 4.4, we need to produce a simulation environment which is capable of computing static physical interactions of grasps efficiently. Thanks to a mature community of researchers in Robotic Grasping, one common tool in simulation of grasps is the GraspIt! [18] simulator, shown in Figure 5.1. It has been designed for researchers with many built-in implementations of some of the most recent methods for simulating, evaluating and planning grasps. The main interesting functionalities for us are:

- Loading geometrically correct and consistent models of commercially available hands.
- Built-in support for eigengrasps of common simulated hands.
- Loading objects from many common formats for meshes, including the Object File Format (.off) used in the Princeton Shape Benchmark [19], a mesh database that we better detail in Section 5.2.
- Moving the robotic hands consistently with their real joints constraints.
- Collision detection in static movements (grasp approach), and computation of the set of contacts with the respective friction cones.
- Management of the reference change matrices in loaded objects. These matrices encode the position and orientations of objects in the simulated worls, as the reference change matrix of an object O maps coordinates expressed in the system of reference of object O to coordinates expressed in a global reference system.
- Built-in support for state of the art grasp robustness metrics.

While being an excellent starting point, still we need to implement extra features to build a data generation tool to collect tuples in the form $< O, p_0, d, \Phi(O, GP(p_0, O), DM(d, O) >$ as described in Section 4.2. In particular, we implement a GraspIt! plugin with the following additional features:

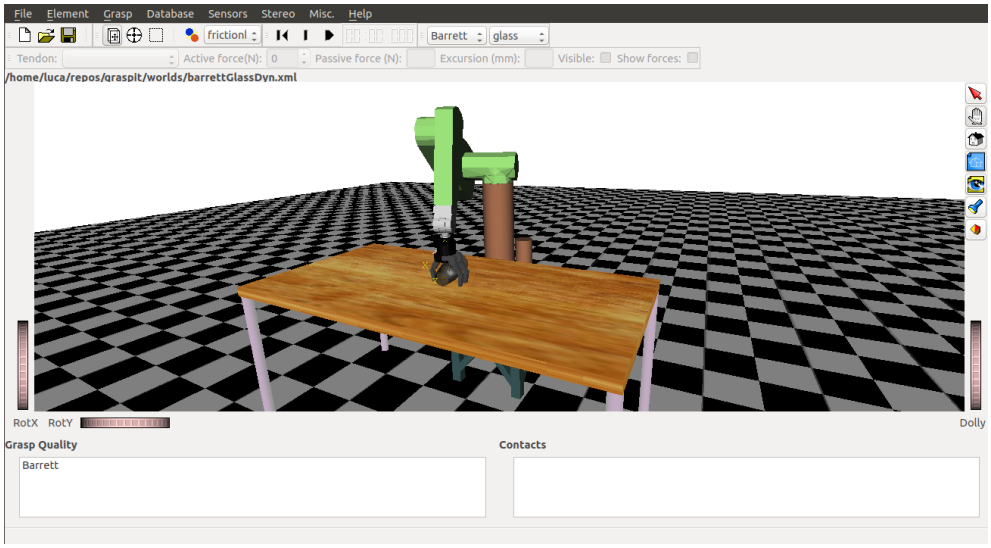


Figure 5.1: A standard world on the GraspIt! GUI

- Grasp parameterization: we need to automatically extract random pregrasps p_0 and use directions d and to set the initial hand posture and position accordingly.
- Grasp actuation: we need to implement our grasping policy GP to effectively produce a simulated grasp from the pregrasps.
- Metrics computation: we need to compute the metrics listed in Section 4.3 from the simulated grasps, by implementing function Φ.
- Data generation: we need to consistently compute and log the metrics ϕ with the corresponding pregrasps p_0 and use directions d in a continuous loop.

We employ the built-in model of the Barrett Hand [10] as it is a very common choice in research, both in simulation and in real experiments. This hand provides the minimum necessary functionality to achieve complex grasps, with three metal fingers, with one active DoF each, while one more DoF, the distal joint of the finger, is active only when the previous joint is locked. One finger is fixed on the palm, while the other two share the same rotational DoF around the base of the wrist, which can only rotate around its axis as well. An eigengrasps decomposition of such degrees of freedom is discussed in Section 5.3.1 and shown in Figures 5.2 and 5.3.

5.2 THE PRINCETON SHAPE BENCHMARK

The Princeton Shape Benchmark [19] is a dataset containing 1814 object meshes in Object File Format collected from the World Wide Web, each

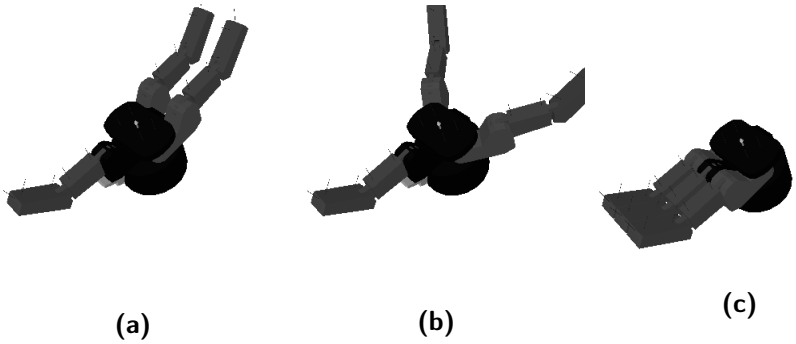


Figure 5.2: Pregrasp degree of freedom (a) set to 0, (b) set to 0.25, (c) set to 1

classified with respect to many semantic criteria. It was proposed in 2004 to represent a single, large reference database to compare new methods for 3D shape matching and classification which were usually compared against different datasets, thus with not completely comparable performances.

We use this dataset as a source of object models as it contains many meshes of objects that can be related to the chosen tasks of cutting, beating and picking. In particular, we look for household objects like hammers, glasses, bottles, kitchen knives; moreover we take objects with similar semantics but different geometries like blades and axes, and some outliers like ice creams and chess pieces. A complete list of the objects involved in our experiments is in Figures 6.3 to 6.6.

We directly load object models from the Princeton Shape Benchmark in Object File Format. This is a very simple mesh description standard, and while GraspIt! provides a built-in parser for this format, we built our own for further uses out of the GraspIt! environment.

5.3 EXTENDING GRASPIT!

As discussed in Section 5.1 we need to implement some specific extra functionalities to the GraspIt! simulator to build our data collection engine. This simulator can be easily extended through plugins written in C++ that have full control over the internal simulation environment.

5.3.1 *Representing the pregrasps and use locations*

Correctly encoding the pregrasps is a sensible step that can potentially bias the grasp search distribution.

Pregrasps need to describe the posture of the hand (thus its degrees of freedom) and its initial position and orientation in the space. A trivial representation of pregrasps would be encoding them directly with the d degrees

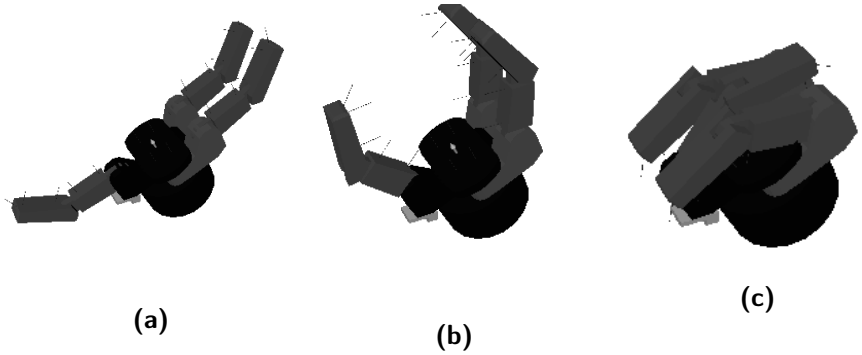


Figure 5.3: Clutch degree of freedom (a) set to 0, (b) set to 0.5, (c) set to 1

of freedom of the hand actuator and 6 more degrees of freedom for translation and orientation in space, thus defining a mapping $\mathbb{R}^{d+6} \mapsto \mathcal{P}_0$.

5.3.1.1 *Efficient representation of pregrasps through eigengrasps*

As a general rule, a representation is as much effective as it is concise, with the lowest possible dimensionality of the parameter vector, while preserving the identity of the corresponding pregrasps. Finding an effective parameterization of grasps is not an easy nor new problem in Robotic Grasping, thus we shall start from state of the art parameterization concepts. As first underlined in [8] the classical effective hand postures that humans perform on everyday objects display a high degree of redundancy, so that the first 2 principal components over the 15 degrees of freedom measured in their experiment explained 80% of common hand postures variability. Such finding has been ported to the field of Robotic Grasping in [23] through the definition of *eigengrasps* as low dimentional bases in the high dimensional DoF space that can parameterize very accurately the effective grasps while counting much less parameters than the number of degrees of freedom.

Jointly with the use of eigengrasps to describe the degrees of freedom of the hand, our selected policy allows for a further reduction in the dimensionality of the pregrasp description for the translational degree of freedom saved in describing the position of the hand in the space, as approaching the object in a straight line makes the approach direction invariant with respect to the policy. We can further divide the eigengrasp parameters into h parameters used to close the grasp and k parameters used to set the initial hand posture: the h eigengrasp parameters used to close the hand are fixed to an open position on the pregrasp, thus they do not contribute to the dimensionality of the pregrasp itself. Therefore, we can describe the initial posture of our grasps with only h parameters and the position of the hand in space with 5 parameters, defining a mapping $\mathbb{R}^{h+5} \mapsto \mathcal{P}_0$ which is way

more efficient than the trivial baseline and scales much better with complex hands.

For our Barrett hand we can use very simple and intuitive eigengrasps: one eigengrasp shown in Figure 5.2 controls the finger shared rotational DoF alone, while one other in Figure 5.3 collectively controls the closure of the fingers. As the latter is fixed to open in pregrasps, we finally use only the former to describe the hand posture in pregrasps.

5.3.1.2 *Parameterization definition*

To control the hand initial location and rotation, we implement two different grasp parameterizations: the first aims at using the minimum number of parameters possible, the second instead aims at eliminating any positional bias in the pregrasp and use location search.

MINIMAL PREGRASP PARAMETERIZATION The minimal parameterization aims at using the minimum possible number of parameters. As we use one degree of freedom for the hand posture and a minimum of five for the location and rotation of the hand, we use exactly six parameters.

The parameterization is obtained by extracting a point P_{off} on a fixed reference plane, the xy plane, (two parameters) in mesh relative coordinates to be used as offset with respect to the center of mass of the target object, and three rotations (roll, pitch, yaw) to be applied to the hand in standard position to determine its approaching angle. The approach direction is determined by the rotated image of the hand front versor, applied on P_{off}. The actual initial hand position is obtained by translating P_{off} backward along the approach direction g_a by a sufficiently long distance as shown in Figure 5.4, while the hand posture is encoded by its single pregrasp DoF as anticipated. Use locations, on the other side, are produced similarly by extracting a use direction (only two rotations in spherical coordinates, as the third is invariant) and intersecting a ray from the center of mass with such direction with the object mesh.

Initial testing with this method highlighted that, although minimal, this encoding requires significant preprocessing computational effort to reproduce the grasp and use location information from the raw parameters, while not providing significant advantages for its brevity. Moreover and more importantly, extracting raw parameters from uniform distributions leads to highly biased distributions of grasps and uses which are concentrated around the center of mass of the object, and more rarely consider grasping or using in farther locations.

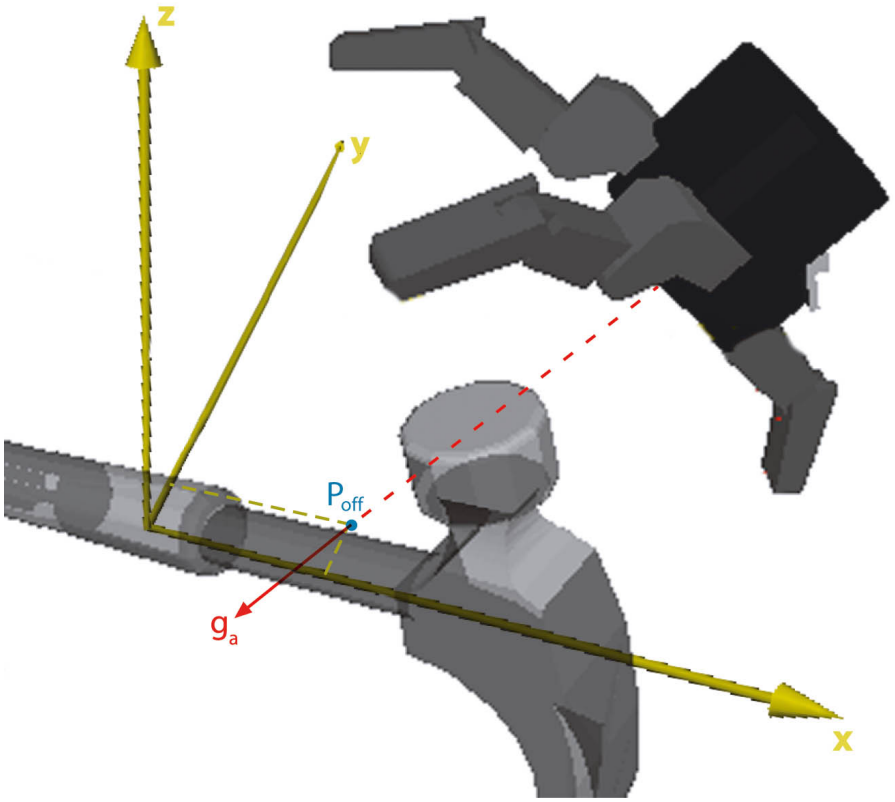


Figure 5.4: Schema of minimal pregrasp parameterization.

UNBIASED PREGRASP PARAMETERIZATION The unbiased parameterization aims at solving the problems highlighted with the first experiments, which are the grasp distribution bias and the high preprocessing cost.

To eliminate the bias around the center of mass, we extract grasp and use locations P_g and P_u uniformly *with respect to the surface of the object mesh* by using state of the art methods [35] which are equivlent to extracting two independent samples from a uniform distribution in the unitary interval and feeding them into the procedure that we describe in detail in Section 6.4.1.1. This guarantees no bias in the extraction of grasp and use locations and, logging the points directly, it needs no preprocessing at all, but it uses three parameters instead of the usual two.

The approach direction is chosen by uniformly extracting unitary quaternions q by a known method from [4], which can be found as a built-in function in the Eigen [32] library, following the formula:

$$h = \left(\sqrt{1-r_1}\sin 2\pi r_2, \sqrt{r_1}\cos 2\pi r_3, \sqrt{r_1}\sin 2\pi r_3, \sqrt{1-r_1}\cos 2\pi r_2\right) \quad (5.1)$$

where r_1, r_2, r_3 are repeatedly extracted independently from a uniform distribution in the unitary interval until the encoded approach direction $\vec{d}_a$ applied on P_g is coming from the outside with sufficient angle, which is equivalent to:

$$\vec{d}_a \cdot \vec{n}_{P_g} \geqslant \tau_g \tag{5.2}$$

being $\vec{n}_{P_g}$ the unitary normal vector in P_g directed inward and τ_g is a threshold that we fix to 0.1 to define a large cone of admitted grasp directions.

This method uses far more parameters than the biased one (three parameters for the grasp location P_g and three more independent parameters for the unitary quaternion q, and again three parameters for the use location P_u), but this disadvantage becomes minor if the grasp direction is implicitly modeled later in the learning problem. On the other side, the quantities produced are describing more directly the translation and rotation of the hand, requiring much less preprocessing time to be used, while granting a sufficiently unbiased exploration of the pregrasp space.

5.3.2 *Performing the grasp*

Performing a grasp from a given pregrasp is equivalent to implementing our grasping policy already described in Section 4.1.1.1 and shown in Figure 4.1.

The pregrasp already sets the hand in its correct initial location and rotation, also setting the value of the one degree of freedom that controls the angle between the two mobile fingers. To maximize the probability to effectively wrap the target object in a wider grasp, we open the hand by setting the clutch eigengrasp to 0 as in Figure 5.3a. We approach the object by advancing on a straight line in the grasp direction until a contact is made, or a threshold distance has passed without any contact, in which case (very rare with the minimal parameterization, impossible with the unbiased one due to the explicit grasp target P_g on the object surface) the grasp is discarded.

Advancement in discrete steps towards the grasp location is already implemented in GraspIt! as a built-in, as well as the detection of any intersection between rigid bodies (hand and object) and the backward interpolation of the motion to obtain a more precise contact. Once one contact is made, the clutch is closed by gradually increasing the clutch eigengrasp until a contact is made with each finger. Every finger advances until it makes a contact *independently* of all the others, thus the final grasp is formally out of the space defined by the eigengrasp basis, which we use only to parameterize *pregrasps* which, given a mesh, are deterministically mapped to a free grasp on it.

5.3.3 *Metrics computation*

Computation of metrics is a core functionality of this work that deserves particular attention. The considered physical metrics, described in Section 4.3.1, are heterogeneous and need both geometrical and static analysis of the mesh and the grasp. Before solving the problem of each metric independently, we make the following assumptions:

- **Rigid body assumption**: we assume that both the grasped body and the hand are rigid bodies, thus they are not subject to macroscopic deformations.
- **Hard contact assumption**: we assume that all contacts are *hard contacts*, thus considering only the transmission of normal forces and tangential forces within a friction cone (see Reference [28] for a detailed explanation of contact models). In particular we always consider objects to be in hard plastic and fingers to be in metal, thus friction cones' aperture follows the plastic-metal static friction coefficient.
- **Arbitrary joint control assumption**: we assume that the hand joints control allows the hand to actuate any combination of forces within the friction cones of its contacts.
- **Beating motion assumption**: we assume that the task of beating is performed by a Barrett Hand by rotation around its wrist, as anticipated and explained in Section 4.3.2.

Taking the ϵ metric aside, which uses the built-in Epsilon metric in GraspIt!, the computed metrics can be divided in two groups: some require geometrical considerations irrespectively of the specific contacts, while others are intrinsically framed as optimization problems on the allowed forces, with eventual external forces.

5.3.3.1 *Geometrical metrics*

The geometrical metrics include the I, δ and U_g. We report here again their description, together with more specific implementation details than in Section 4.3.1.

ROTATIONAL INERTIA ($I \in \mathbb{R}$) I quantifies the rotational inertia around the axis of rotation of the wrist of the hand assuming a unitary density of the object and assuming the hand to be integral with the whole object. It does not take into account the mass of the hand itself. The implementation of this metric relies on the code from [22] to compute the inertial matrix of a body described by a mesh, then the actual value of inertia with respect to

the wrist rotation axis is computed, taking into consideration the distance of the axis with respect to the center of mass.

MOMENTUM DISCHARGE EFFICIENCY ($\delta \in \mathbb{R}$) δ quantifies the efficiency of discharging the rotational inertia of the wrist on the object use location. It quantifies the alignment between the inertial torque τ_I and the torque $\tilde{\tau}_u$ generated by a force aligned with the use location normal vector towards the inside of the object surface. It is computed as follows:

$$\delta = max(0, \frac{\tilde{\tau}_u}{||\tilde{\tau}_u||} \cdot \frac{\tau_I}{||\tau_I||}) \tag{5.3}$$

USE LOCAL GEOMETRY ($U_g \in \mathbb{R}$) U_g describes how much the use location has the shape of an edge. It is obtained by fitting a quadratic function on the vertices of the triangles near the use location (including all the triangles that share at least one vertex with the triangle where the use location lies) and extracting the eigenvalues of the hessian matrix of such quadratic function. The two eigenvalues λ_1 and λ_2 are the two principal component curvatures, so we quantify an edge with the expression $(\lambda_1 - \lambda_2)^2$ to identify locations with a great difference in local curvatures. Both the idea and the algorithm of fitting a quadratic surface onto the mesh are taken from [24] who uses the same technique to estimate the contact surface in soft contact modeling. The code is taken from the GraspIt! implementation of the soft finger model.

5.3.3.2 *Optimization metrics*

Optimization metrics include E_h, E_i and U_τ. They are intrinsically formulated as either minimization or maximization problems onto the contact forces within the friction cone constraints. We frame such problems as Linear Programming problems, taking contact forces as variables, friction cones and equilibria as constraints with possibly either controllable or fixed external forces and torques included.

We produce a general linear grasp optimizer built on top of the CGAL [27] optimization library to setup the linear problems encoding a set of contact forces. The set of contacts $C_1 \dots C_{n_c}$ with external fixed wrenches $\bar{W}_1 \dots \bar{W}_k$ and external variable wrenches $W_1 \dots W_v$ are encoded in the following constraints:

$$\sum_{j=1}^{8} t_{i,j} \leqslant \mu_s n_i, \qquad i = 1 \dots n_c$$

$$\sum_{i=1}^{n_c} M_i \begin{bmatrix} t_{i,1} \\ \vdots \\ t_{i,8} \\ n_i \end{bmatrix} + \sum_{j=1}^{k} \bar{W}_j + \sum_{l=1}^{v} W_l w_l = 0,$$

$$0 \leqslant t_{i,j}, \qquad i = 1 \dots n_c, j = 1 \dots 8$$

$$0 \leqslant n_i \leqslant \tau_N, \qquad i = 1 \dots n_c$$

$$0 \leqslant w_l, \qquad l = 1 \dots v$$

where variables n_i are the length of the used normal component in each contact C_i, variables $t_{i,j}$ are the tangential components of the friction cone of contact C_i along the direction j as shown in Figure 5.5 (here we consider 8 directions equally distributed on a unitary circle as a discrete approximation of friction cones), w_l are the variables associated with the usage of variable wrenches W_l. Matrix M_i maps the coefficient vector $[\, t_{i,1} \dots t_{i,8}\ n_i \,]^T$ of a contact C_i to the corresponding wrenches in the global system of reference. Matrix M_i is composed by a fixed cone encoding matrix Γ that maps the coefficient vector $[\, t_{i,1} \dots t_{i,8}\ n_i \,]^T$ into a force vector in contact coordinates within the friction cone according to Figure 5.5, and a reference change matrix R_i to switch from contact to the global reference system.

HAND EFFORT ON IMPACT ($E_i \in \mathbb{R}$) describes the effort of the hand to balance the impact forces after a rotation around the wrist. It assumes a fixed average inertial torque in a small Δt during the impact which is directly proportional to I and a free contact force on the use location towards the normal direction.

Let P_u be the point of use with its normal $\vec{n}_{P_u}$, O_{COG} the center of gravity of the object, $\vec{z}_g$ the grasp approach direction versor, then the problem is formulated as a grasp LP with the following parameters:

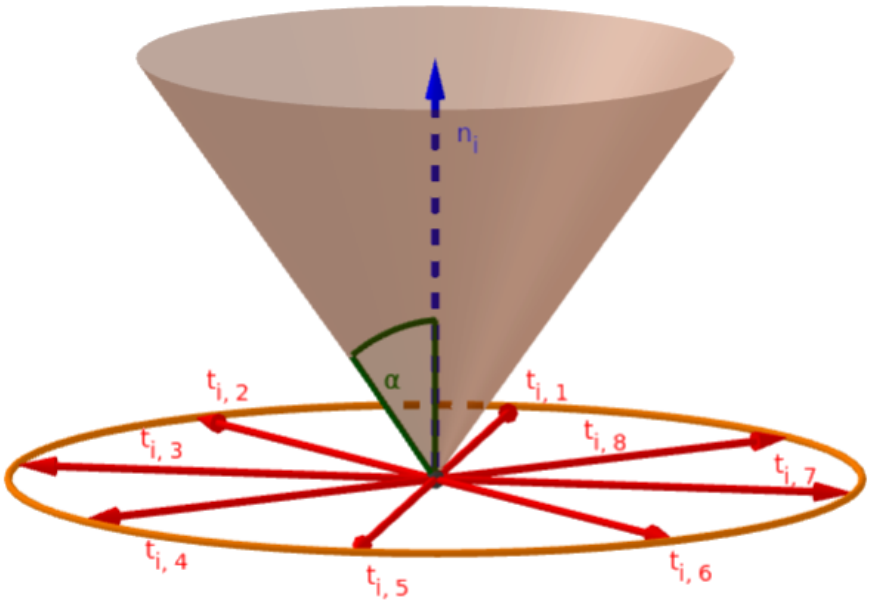


Figure 5.5: Friction cone parameterization with aperture $\alpha = \tan^{-1}(\mu_s)$ to describe contact C_i in the LP solver.

$$k = 1 \qquad v = 1 \qquad \tau_N = +\infty$$

$$\bar{W}_1 = \begin{bmatrix} \mathbf{0} \\ I\vec{z}_g \end{bmatrix}$$

$$W_1 = \begin{bmatrix} -\vec{n}_{P_u} \\ -(P_u - O_{COG}) \times \vec{n}_{P_u} \end{bmatrix}$$

With objective:

$$\text{minimize} \quad \sum_{i=1}^{n_c} n_i \tag{5.4}$$

HAND EFFORT ON HOLD ($E_h \in \mathbb{R}^6$) is a vector of six independent values which quantify the hand effort to balance a different gravity vector for each component of E_h. The hand effort is the minimum sum of all contact forces constrained to the contact friction cones that balance a given unitary force of gravity, ∞ if such problem is unfeasible. The six gravity vectors chosen are

aligned with the three coordinate axes (once in the same direction, once opposite) of the object mesh as all meshes that we used in the Princeton Shape Benchmark have been designed by humans that gave a semantic meaning to the coordinate axes directions, aligning them with the sides of objects with an implicit notion of up, down, left, right, front and back. We use such design bias as a prior of where the gravity is more likely to be aligned, as objects are usually lying on one of such sides.

In particular, if $\vec{g}$ is the chosen gravity vector, then the problem is formulated as a grasp LP with the following parameters:

$$k = 1 \qquad \nu = 0 \qquad \tau_N = +\infty$$
$$\bar{W}_1 = \begin{bmatrix} \vec{g} \\ \mathbf{0} \end{bmatrix}$$

With objective:

$$\text{minimize} \quad \sum_{i=1}^{n_c} n_i \tag{5.5}$$

FORCE TRANSMITTED TO USE ($U_\tau \in \mathbb{R}$) quantifies the force that can be transmitted to the use location using constrained contact forces assuming that the object is aligned with the gravity vector parallel to the normal of the use location. It assumes also that all contact forces are constrained by their friction cones and have unitary maximum normal forces. It takes the value of the maximum force on the use location towards the use location normal guaranteeing static conditions.

Let P_u be the point of use with its normal $\vec{n}_{P_u}$, O_{COG} the center of gravity of the object, M the estimated mass of the object (assuming uniform material, it is proportional to its volume), then the problem is formulated as a grasp LP with the following parameters:

$$k = 1 \qquad \nu = 1 \qquad \tau_N = 1$$
$$\bar{W}_1 = \begin{bmatrix} M\vec{n}_{P_u} \\ \mathbf{0} \end{bmatrix}$$
$$W_1 = \begin{bmatrix} -\vec{n}_{P_u} \\ -(P_u - O_{COG}) \times \vec{n}_{P_u} \end{bmatrix}$$

With objective:

$$\text{maximize} \quad w_1 \tag{5.6}$$

5.3.4 *Data generation loop*

Having the tools for grasp simulation and for metrics evaluation, we require an effective data generation loop to maximize the amount of quality data that we log. We initially implemented a trivial loop that just performs a fixed number of random simulations, hypothesizing a fixed number of grasps and a fixed number of use locations for each grasp, computing all metrics and logging all results. This initial solution allowed us to test and understand the bottlenecks in the data generation process and which grasps bring differentially more valuable information to the dataset.

After experimental tests we determined that the main bottleneck of our system was in the LP solver for the metrics computation, initially accounting for the vast majority of computational time, and the great majority of LP metric runs was due to the multiple use locations. At the same time most of the random grasps (one over twenty) were not even stable, making complex metrics not meaningful to be computed.

We greatly optimized effective data collection by evaluating robustness as early as possible to know when a grasp is worth evaluating more or being discarded. Unstable grasps are not evaluated along multiple points of use, as their value lies only in detecting their lack of force closure, while robust grasps are fully evaluated in multiple use locations. Moreover, as unstable grasps provide the great majority of random examples with any simple randomization policy, they are going to build an unbalanced dataset. We directly balance the produced dataset by counting the number of stable grasps produced and logging unstable grasps only when they equal the number of stable grasps, thus saving further computational time. With the described policy, the running time spent is more balanced between searching for grasps and evaluating metrics for stable and interesting grasps.

5.3.5 *Known problems*

The following is the list of the known problems:

- The program occasionally freezes on the LP problem resolution during the computation on metrics. This problem is more frequent on some meshes, suggesting that it may be caused by degenerate cases that are randomly picked up. Tests show that a pregrasp that causes a freeze on a specific metric computation deterministically produces the freezing on the same LP, however even a very slight perturbation (10^{-5} rela-

tive perturbation on any single parameter of the pregrasp) solves the problem.

- The meshes in the Princeton Shape Benchmark models often approximate sharp edges with either flat closed surfaces or open plains. Such approximations usually make blade edges difficult to be detected by the U_9 metric in simulation.

5.4 AUTOMATED DATA GENERATION

Data generation should be as much scalable as possible. This is translated into two main requirements: it needs to be parallelizable on multi-core machines and it is requires little to no interaction and maintenance. Both these features are not readily available with GraspIt! as it is a research simulator which has not been developed specifically to grant parallelism and reliability, therefore we satisfy these requirements by wrapping the main simulator for data collection with an automated process management tool that we designed on purpose.

5.4.1 *Parallelism*

In this context scalability comes with the ability to parallelize data collection, both in the process and data logging. The actual simulation process is intrinsically highly parallel as single simulations are independently executed for randomized grasp parameters, while logging of results can easily be done on different files that are merged later, as explained in Section 5.4.3. The main obstacle is the fact that the GraspIt! simulator is not designed to be executed in a multithreaded environment, having only partial and experimental support to multithreading. Therefore, we step back to a multiprocessing solution: our process management tool monitors available data and launches different independent processes collecting data for a limited period of time on different objects, logging on different target files. This method allows to easily monitor and balance data collection, dynamically allocating more cores to the collection of objects which require more effort to collect and measure valid grasps.

5.4.2 *Reliability*

A highly parallel infrastructure is not sufficient to scale data collection if independent processes are unreliable and need frequent human intervention. As underlined in Section 5.3.5, the simulation process alone is subject to unpredictable freezes during data collection, which requires a hard restart of the process.

To grant reliability we inserted a timer into the simulator to terminate the process to free the resources if any grasp metric computation takes much more than the observed regular maximum (which is two seconds on the machine we used, with a timer of ten seconds). At the same time the process management tool monitors the running processes and restarts a new process whenever one dies for any reason. The new process is started on the object that counts the least amount of available data, to balance the dataset.

5.4.3 *Data merging*

The output of our wrapped data collection system is a set of text files where each line logs a pregrasp, a use location and all the relative measured metrics. Each file is relative to a single object mesh from the Princeton Shape Benchmark [19], and multiple files relative to the same mesh exist to ensure that multiple processes do not have to synchronize to simulate on the same mesh.

The ideal shape of the dataset is a zipped archive with a neat structure: as we separate training, validation and test sets based on the target objects, we want one single data file for each object mesh. For this reason a last post-processing script is run to merge all available data samples of each mesh into a single data file, finally zipping all data files into an archive.

5.5 LEARNING FROM DATA

The long term objective of this research is achieving affordance inference from vision on novel objects, thus we address the applicability of our framework through learning from the data we collected.

5.5.1 *Data preprocessing*

While the raw data are in the form of tuples of values encoding grasps, use locations and metrics, the models described in Section 6.1 expect high dimensional data as it comes from range cameras: a data preprocessing pipeline is necessary to connect data with the available models, decoding the pregrasp information and synthesizing depth images and point clouds.

As pregrasps are encoded into implicit parameters, they need to be decoded into a directly useful form as the explicit reference change matrix to switch from world to grasp coordinates. This step is potentially performance sensitive as it needs to be performed for every single grasp ready from data: a first implementation using the pyquaternion package resulted into a bottleneck of the whole data preprocessing pipeline, therefore we base our final implementation upon the numpy-quaternion package, as it is based on a

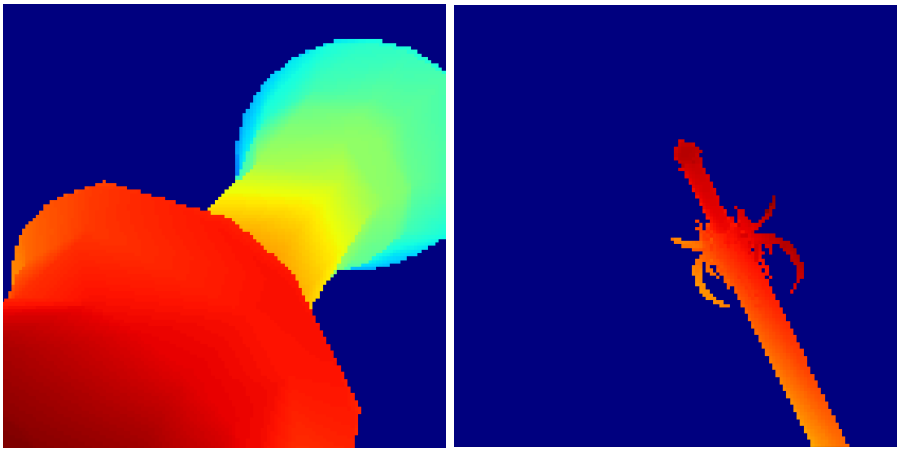


Figure 5.6: Example synthesized depth images. Red is near, blue is far.

C++ implementation built onto numpy. We considered the hypothesis of providing a GPU implementation to greatly speed up this step, but the current solution does not take a significant enough portion of the preprocessing time anymore.

The actual implementation highly depends on the encoding used into the data, as explained in Section 5.3.1. The minimal parameterization requires performing many subsequent rotations of the canonical reference system to decode the spherical coordinates, while the unbiased parameterization, built upon the experience of the previous one, has the intrinsic advantage of providing directly the quaternion encoding the destination reference change, thus it only needs to decode the quaternion into its equivalent transformation matrix form.

The next step of the preprocessing phase is generating the range images of the object from the grasp reference. As this is clearly the most computational intensive task of the whole preprocessing pipeline, we spent much effort in its optimization as it is the main reason for the preprocessing time.

We provide a GPU implementation based on OpenGL: the mesh vertices are directly loaded to the GPU memory by buffers, the camera reference is moved according to the grasp reference, mirroring the z axis to match the looking direction with the grasp direction, and the OpenGL pipeline is executed with standard shaders, backface culling and z-buffering. We use a perspective projection with parameters inspired to the Kinekt 2 depth camera (70x60° field of view angle): we generate subsampled images of resolution 128×128 using a 60° field of view taken from an object-length distance from the center of the object. The z-buffer is extracted and the normalized depth image is obtained from it according to the following formula:

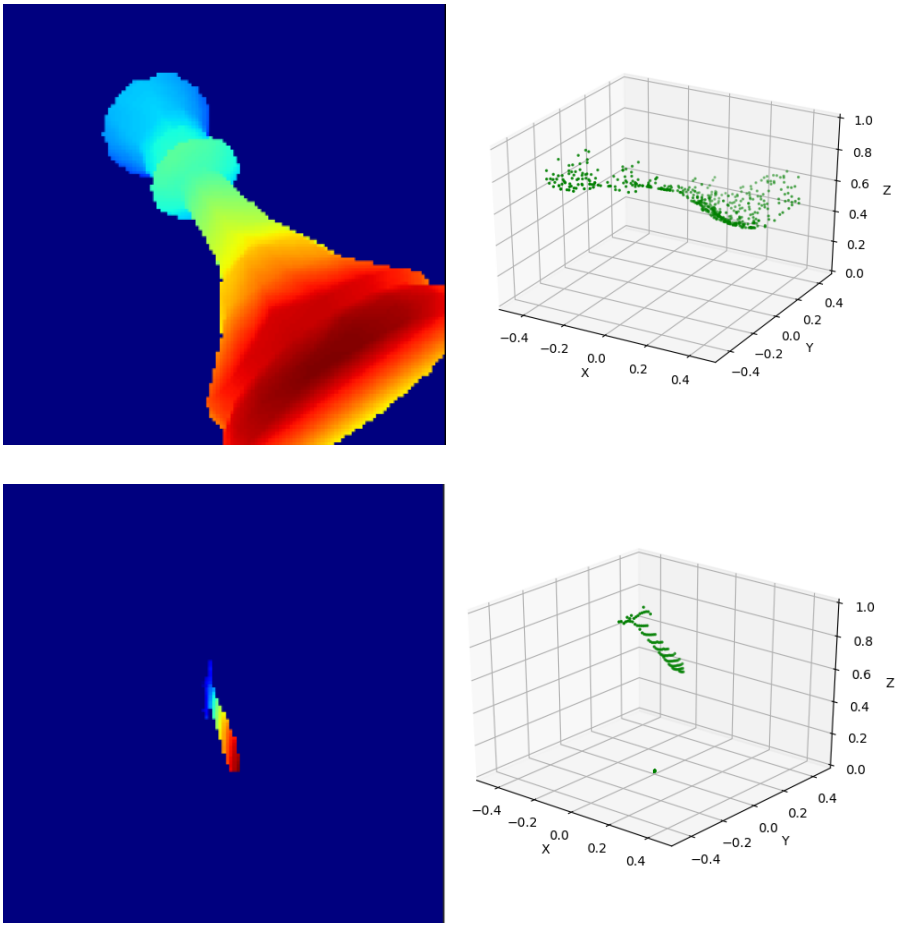


Figure 5.7: Sample synthesized point clouds with the corresponding original images.

$$d(u,v) = 2\frac{Z_{far}Z_{near}}{Z_{far} + Z_{near} - (Z_{far} - Z_{near})B(u,v)},$$
$$D(u,v) = \frac{1}{Z_{far} - Z_{near}}(d(u,v) - Z_{near}),$$

where B is the z-buffer, Z_{near} is the near plane, Z_{far} is the far plane, d is the actual depth map, and D is the normalized depth map used as input for the networks. Examples of synthesized images are shown in Figure 5.6.

Point clouds hold the same information as a depth image, thus they can be generated directly from them. If (C_X, C_Y) are the pixel coordinates of the virtual camera center and f_X and f_Y its focal lengths, then each pixel (u,v) is mapped to a point (X,Y,Z) as follows:

$$Z = d(u, v)$$
$$X = \frac{(u - C_X)Z}{f_X}$$
$$Y = \frac{(v - C_Y)Z}{f_Y}$$

For performance reasons this function is not implemented in pure python, but it is based on the Open3D [63] package which is a wrapper for the C++ library.

The produced point cloud incorporates both the object and the planar background at Z_{far}, therefore it is cleaned by eliminating all points which are too near to Z_{far}. This cleaning procedure makes all meshes to have a variable number of points, which is incompatible with batch learning as it needs to fill precise tensors with batches of point clouds that must have the same number of points. Therefore, we fix a number of points that each point cloud must have: if after the cleaning it contains too many points, the needed number of random points is dropped; if it contains fewer points then the needed amount of points are added all in $(0, 0, 0)$. Sample point clouds with their original range images are shown in Figure 5.7. This last step of point cloud cleaning is not performed in the specific case of the Local PointNet described in Section 6.1.3 as it needs to map each point to its corresponding pixel position in the original depth image.

5.5.2 *Training setup*

The whole training infrastructure has been written with the Keras package with Tensorflow backend, with additional integrations directly in Tensorflow, and Tensorboard logging to monitor the training.

We train using the Adam Optimizer [41] with a learning rate λ that decays according to the formula:

$$\lambda_t = \frac{\lambda_0}{1 + t} \tag{5.7}$$

Where t is the epoch counter from 0 to t_{max}. We use dropout [43] and early stopping to reduce overfitting [11]. We use the maximum batch size possible, which depends on the memory requirements of the trained model and on the hardware available for training, which is run on GPU.

As the memory requirements to train with millions of synthesized images of point clouds are extremely high, we store the images in RAM in a quantized form to byte precision, to be restored to full precision type only for the batch forward and backward passes. Such practice determines a loss in precision on the input data at training time that we do not reproduce

when testing, but experimental evidence (no significant difference between training losses on the reduced and full precision samples) suggests that this practice does not have any true effect on the learning process of the models. We think that this finding is very reasonable as we expect the geometrical pattern to be learned to lie in aggregate high level structures rather than in fine micropatterns two orders of magnitude smaller than the maximal distance (which is the precision still granted by bytes).

6

EXPERIMENTAL VALIDATION

To assess the feasibility of the proposed framework we performed a set of experiments focused on the tasks selected in Section 4.3, aiming at:

1. **Validating the framework** by showing that $\arg\max_{G,U} \tilde{F}_T(\Phi(O, G, U))$, based on grasp metrics Φ, for some selected tasks T, provides grasps and use locations that are semantically meaningful with respect to the semantics of task $T \in \{\texttt{beat}, \texttt{pick}, \texttt{cut}\}$. This step questions the intrinsic validity of the theoretical framework in a perfect information setting, assessing qualitatively that the best grasps obtained by exhaustive search over the collected data are suitable for the selected tasks. The quality obtained at this step can be considered an upper bound of what an applied system can achieve in a real environment with partial information for two strong reasons: firstly in this case we consider to have full information about the object, secondly the models for partial information are expected to be trained on these same data, thus they are not expected to be able to surpass the quality level which is intrinsic in data.

2. **Assessing the feasibility of learning** a model $\mathcal{M}^\Phi$ that can infer basic grasp metrics ϕ from partial information about a target object as detailed in Section 4.2. This step questions the applicability of such framework to a context of uncertainty and partial information, which is typical of real contexts. Currently, our validation is limited to the quantitative evaluation of $\mathcal{M}^\Phi_C$ and $\mathcal{M}^\Phi_R$ learning modules, and the qualitative evaluation of the integrated model $\mathcal{M}^\Phi$, while no test has been performed with real robots for now.

All experiments have been run on a machine with an Intel Xeon E5-2630 v4 CPU with 40 cores, 8 Nvidia GeForce 1080 GPUs and 256GB of RAM, but for the experiments only 2 GPUs and 20 to 25 cores have been used.

6.1 STRUCTURING THE LEARNING PROBLEM

As anticipated in Section 4.2, we want to learn a model $\mathcal{M}^\Phi$ from a dataset of tuples $< O, p_0, d, \Phi(O, GP(p_0, O), DM(d, O)) >$, where $\mathcal{M}^\Phi$ is divided into a first classifier $\mathcal{M}^\Phi_C$ that filters stable grasps worth considering, and a regressor $\mathcal{M}^\Phi_R$ that infers the basic metrics $\Phi(O, GP(p_0, O), DM(d, O))$ from filtered grasps.

The depth maps $D(p_0, O)$ are synthesized with camera-in-hand perspective to encode both local information about the object geometry and implicit information about the grasp location and orientation. The DoF information is considered as an extra input to the model together with the depth image, to complete the description of the pregrasp p_0.

The classifier $\mathcal{M}_C^\Phi$ performs a binary classification, thus its output is always a value in the range $[0, 1]$ which represents the probability of having a stable grasp. As we require positively filtered grasps to be viable for the execution of some task, requiring strict force closure ($\epsilon > 0$) would be not enough, thus we employ a custom and more restrictive definition of stable grasp that involves a minimum general robustness τ_ϵ and a maximum overall effort to hold the object against gravity τ_{E_h}:

$$\epsilon > \tau_\epsilon \quad \wedge \quad \sum_{i=1}^{6} E_h[i] < \tau_{E_h} \tag{6.1}$$

where empirically we set $\tau_\epsilon = 0.15$, $\tau_{E_h} = 250$.

Since we use a custom definition of stable grasp which incorporates more stringent and complex requirements than the regular stability, for the sake of formal correctness from now on we refer to them as viable grasps, which are the positive class of our classification network $\mathcal{M}_C^\Phi$.

We provide several models for both the classifier $\mathcal{M}_C^\Phi$ and the regressor $\mathcal{M}_R^\Phi$ which come from variations of the Xception Convolutional Neural Network [53] model and a PointNet [56] model.

6.1.1 *Convolutional Neural Network model*

We tested some configurations of Convolutional Neural Networks for both the regressor and the classifier networks. All of them take a normalized depth image as input and are conditioned with the DoF parameter to finally output a single score which is either the classification label or a single metric ϕ for regression.

As Convolutional Neural Networks have attracted much research attention in the last years and were greatly improved both in performance and in efficiency, we take one very recently proposed architecture as starting point. The base specific architecture we use is the Xception architecture [53], shown in Figure 3.9. This architecture takes to the extreme the Inception [49] assumpion of the separation between pointwise and spacial correlations in performing convolutions efficiently using its parameters only with depthwise separable convolutions (channel-wise independent convolutions followed by 1×1 convolutions). Moreover, it includes the residual connections [51] introduced into the Inception family with Inception-ResNet [59] to allow effective optimization along its 36 convolutional layers. A much more in depth expla-

nation of the techniques and assumptions used in this architecture have been explained in Section 3.2.3.

The following are the two main variations considered:

- **Late fusion CNN**: the DoF input is concatenated to the encoding of the image *after* the convolutional segment of the network, within the fully connected layers section of the exit flow in Figure 3.9, as depicted in the upper side of Figure 6.1. Such technique is very simple, but the local geometry is encoded independently of the DoF input, which may be a relevant element to consider to give more importance to different features of the object geometry. We call this variant with the codename of CNNL.
- **Early fusion CNN**: the DoF input is replicated into an image channel and concatenated directly to the image input, to be considered early in the network encoder, as depicted in the lower side of Figure 6.1. This technique has the advantage of conditioning the image encoding features with the secondary input, however the receptive field at the point of first encoding the secondary input itself is very limited with respect to the late fusion alternative. To overcome this limitation, we propose the input DoF parameter again to the fully connected segment of the network, exactly as in the late fusion alternative. We call this variant with the codename of CNNE.

The possibility of performing transfer learning from pretrained CNNs is an extension of the late fusion alternative rather than an alternative by itself. This is clearly not compatible with the early fusion as, to the best of our knowledge, there are no early fusion CNNs with the conditioning input having sufficiently strong similarities with our DoF parameter. Moreover, as shown in [57], transfer learning from RGB CNNs is detrimental for depth-based CNNs, thus we require models trained with depth images for relevant tasks. Such alternatve has been considered but not implemented yet due to the lack of available suitable pretrained models for our task.

6.1.2 *PointNet model*

The PointNet model [56] shown in Figure 3.10 has been designed specifically to effectively process unordered sets of points and tested on many tasks involving inputs in the form of point clouds, such as point cloud classification, part segmentation or semantic segmentation. We skip a detailed explanation of the PointNet architecture and its underlying assumptions as it can be found in Section 3.3 of this thesis and in [56].

Considering the assumptions on which the PointNet is based in our specific case, having invariance under permutations and capturing the geometrical interactions among points is extremely useful, but the invariance under

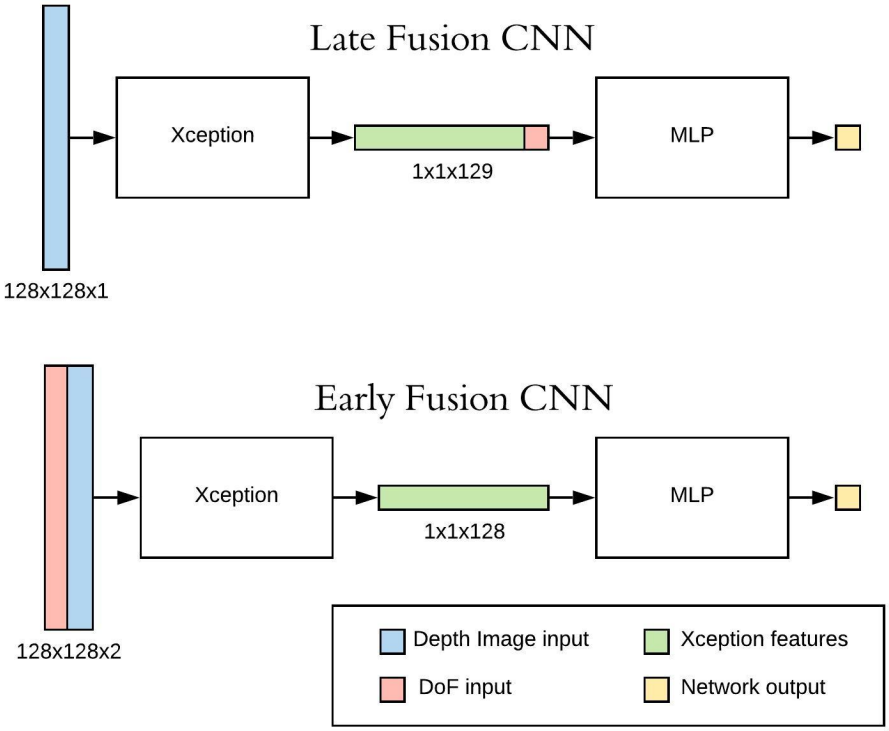


Figure 6.1: Schema of early and late fusion variations on the Xception CNN architecture.

transformations is detrimental as the reference system of the point cloud is meaningful in our case. Such invariance, however, is not intrinsically built in the architecture of the PointNet model, instead the model has the potential to learn it when relevant.

Under these considerations, the PointNet model is a promising model for our task, where the actual need for the transformation net components is uncertain; therefore we test four variations of the PointNet architecture:

- **Full PointNet**: exactly as the classification PointNet described in the original paper, with an additional late fusion of the DoF. As the PointNet embeds each point of the input into a feature vector independently, finally merging all point embeddings by a max pooling, we do not consider that early fusion of the DoF input may be of any use for conditioning each point embedding considered independently of the others. Testing also the early fusion alternative for all PointNet variants would lengthen considerably the list of models under test, therefore for now we reserve the validation of this assumption for future works. We call this variant with the codename of PNFULL.

- **PointNet with Features transform Only**: as the invariance under system of reference is obtained through the initial transformation net (T-net) that infers a 3×3 transformation matrix to learn the invariant, we directly cut this section of the PointNet to fix the refernce change to identity, which we assume to be the correct prior reference change. We call this variant with the codename of PNFO.

- **PointNet with Points transform Only**: as the feature transform net makes up for most of the complexity of the PointNet model, we test a variant in which this transform net is removed (equivalently to fixed to identity) as a strong regularization of the PointNet. We call this variant with the codename of PNPO.

- **Slim PointNet**: we test a combination of the two elements introduced in the PNFO and PNPO variants of the PointNet: we remove both transform nets, producing a much lighter and simpler model. The resulting network with no transform is perfectly equivalent to embedding each point with a MLP, performing max-pooling of all point embeddings, and finally producing the output with an MLP conditioned with the DoF input. We call this variant with the codename of PNSLIM.

6.1.3 *Local PointNet model*

The PointNet architecture, discussed in Section 6.1.2, combines the two assumptions of unordered set of points in input and interaction among points

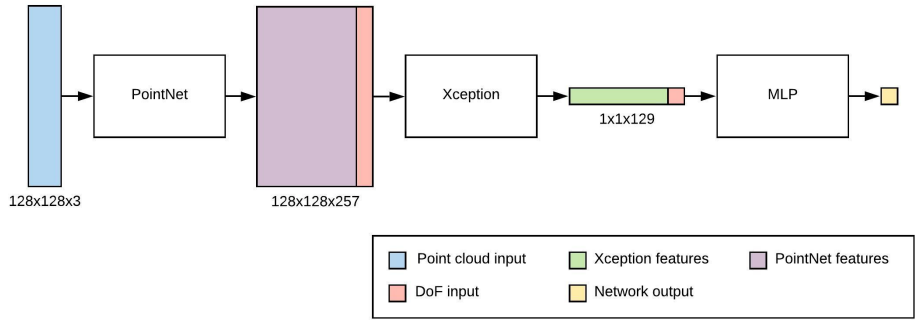


Figure 6.2: Schema of the Local PointNet architecture.

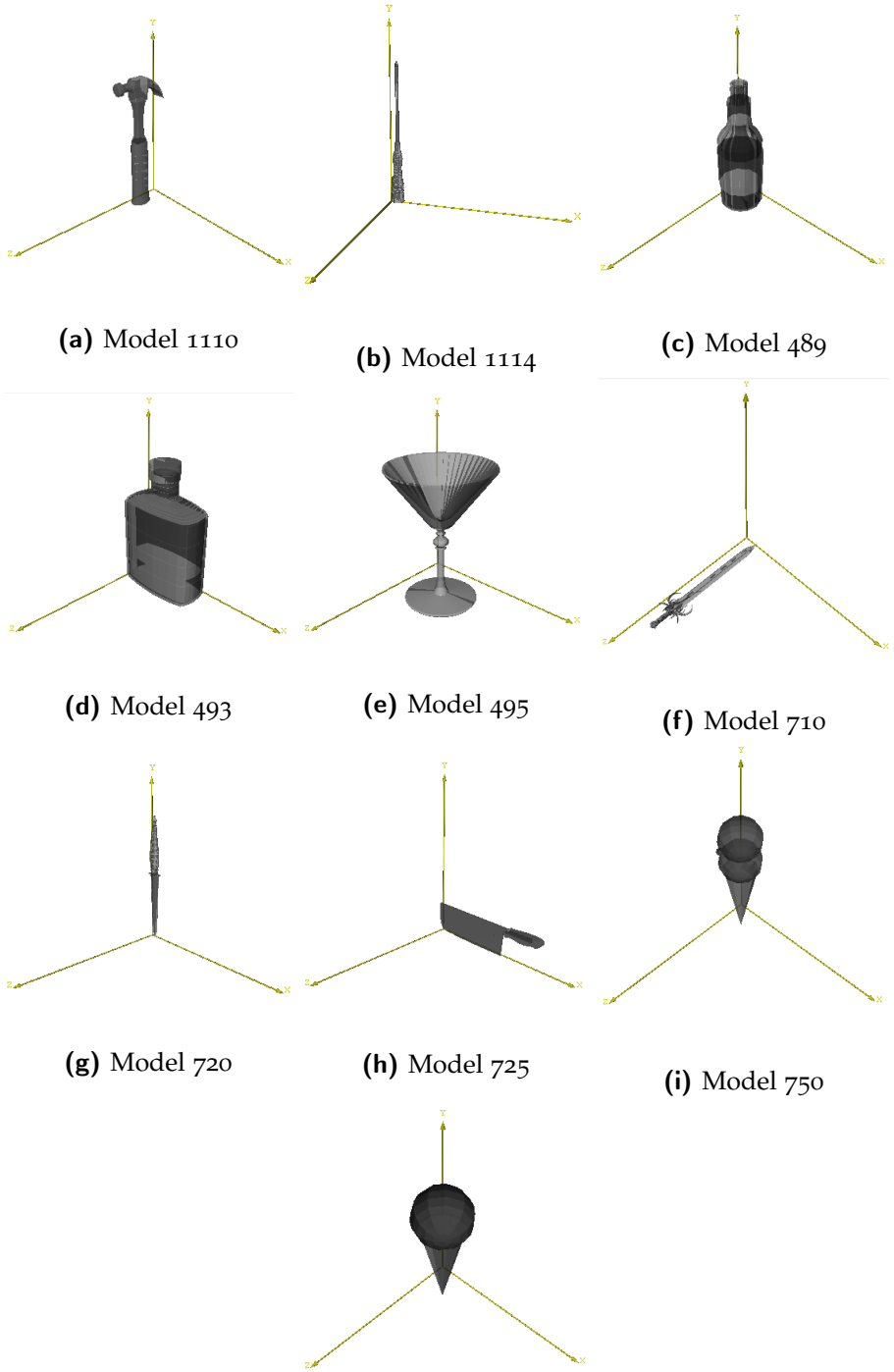
by modeling global interactions with global max pooling, as no specific couple of points is supposed to be more likely correlated than others a priori.

As we expect to detect simple geometrical patterns within our point cloud, we expect to have stronger correlations between nearby points in the point cloud, which we want to capture. On the other side, classical image processing convolutional architectures are specialized in capturing precisely these local correlations between nearby pixels to build up on subsequently higher level representations of the input image.

Following these observations, we propose the Local PointNet architecture, with the objective of introducing the hierarchical encoding of convolutional neural networks to point clouds to capture local geometrical patterns. The Local PointNet is constituted by a base full PointNet that encodes each point *independently* while considering global features only within the transformation nets, as discussed in Section 3.3. The points are fed in the same order as they appear within the depth image that generated them, so that this order (which is assumed and used as main indicator of locality) is preserved by the PointNet that only produces a feature encoding of length 256 of each single point. The feature encodings are presented into an image in the same shape and order as in their original depth image, and passed through a CNN that hierarchically captures local to global patterns. We use the Xception [53] architecture as CNN architecture for the same reasons explained in Section 6.1.1, introducing the DoF conditioning both between the PointNet and the Xception and after on the Xception features, as shown in Figure 6.2. We call this variant with the codename of LPN.

6.2 DIRECT OPTIMIZATION OVER COLLECTED DATA

Direct optimization over collected data has been implemented as a trivial extensive search for the collected data samples whose metrics vector ϕ maximizes a given affordance function $\tilde{F}_T$.



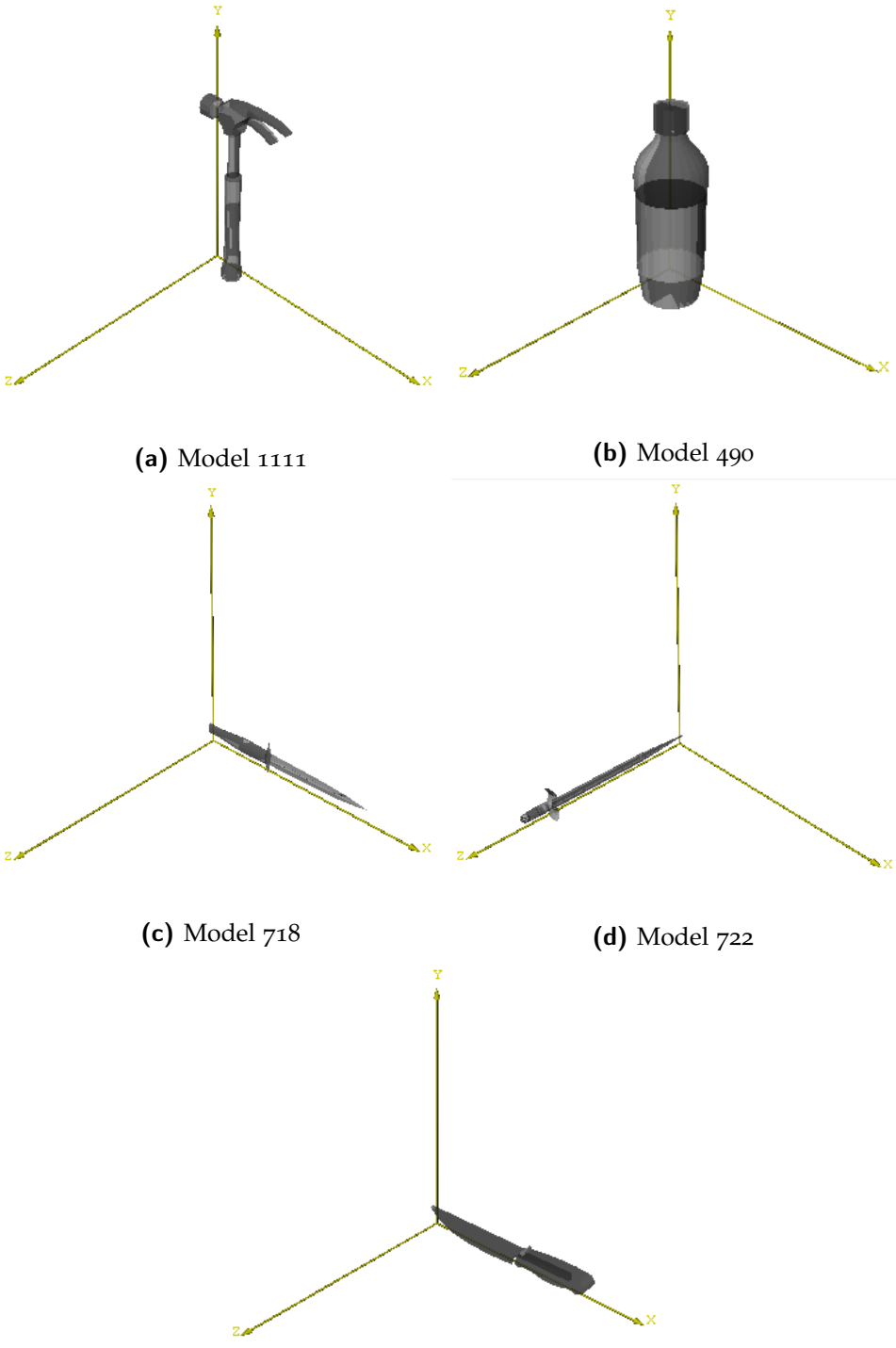
(a) Model 1110 **(b)** Model 1114 **(c)** Model 489

(d) Model 493 **(e)** Model 495 **(f)** Model 710

(g) Model 720 **(h)** Model 725 **(i)** Model 750

(j) Model 758

Figure 6.3: Objects from the PSB included in the training set.



(a) Model 1111

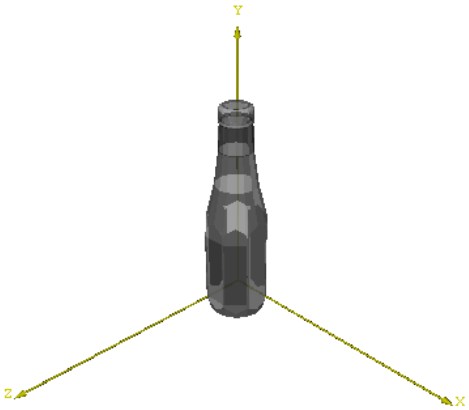
(b) Model 490

(c) Model 718

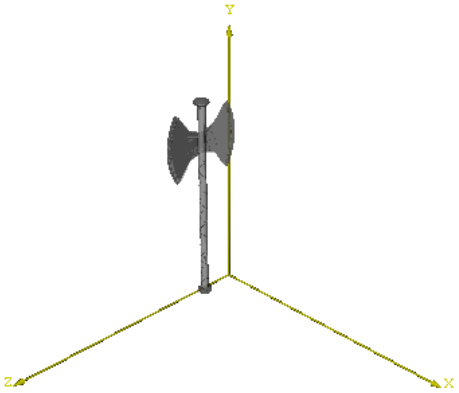
(d) Model 722

(e) Model 724

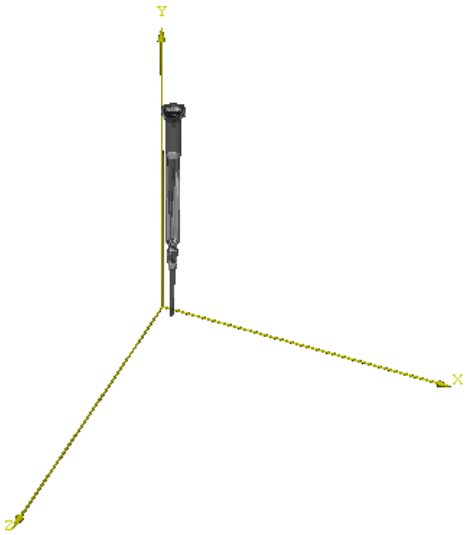
Figure 6.4: Objects from the PSB included in the validation set.



(a) Model 482

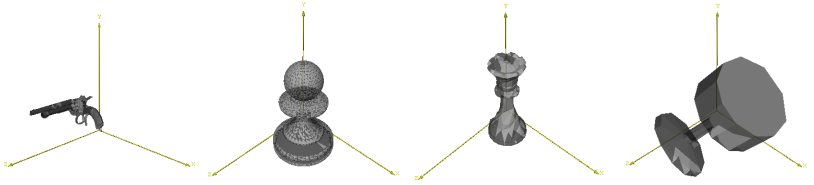


(b) Model 730



(c) Model 1116

Figure 6.5: Objects from the PSB included in the test set.



(a) Model 660 **(b)** Model 1610 **(c)** Model 1614 **(d)** Model 500

Figure 6.6: Objects from the PSB not included in any learning set.

Round	core days	Samples[M]	GGS[M]	UGG[M]	UGG/obj[K]
1	350	400	20	1.25	56.82
2	280	97	91.3	5.7	259

Table 6.1: Data collection rounds description.

The results of direct optimization differ based on the randomization policy used, which depends on the parameterization, described in Section 5.3.1, and on the data generation loop, described in Section 5.3.4. We passed through two rounds of data collection over all the 22 objects shown in Figures 6.3 6.4 6.5 6.6; we summarize the results in Table 6.1, where we report for each data collection round the core days as a measure of runtime (number of cores multiplied by number of days), the number of samples collected, the Good Grasp Samples count (GGS), Unique Good Grasp samples (UGG) and UGG for each single object:

1. The first round lasted two weeks on 25 cores, and collected data using the minimal parameterization with the trivial data generation loop. For this reason it collected many grasp hypotheses, but only a small amount of them is differentially very significant: the great majority of grasps are not viable, bringing no information for the metric regression and constituting a highly unbalanced dataset for the classification task.

2. The second round lasted two weeks on 20 cores, and collected data using the unbiased parameterization with the optimized data generation loop. Even though the overall number of logged samples per unit of computational time is reduced by a factor of 3.3, the collection of significant (viable) grasp samples per unit of computational time is increased by a factor of 5.7.

We represent resulting grasps using the builtin GraspIt! GUI: the target grasp to be represented is performed within the simulated environment, then it is rendered on the Graphical User Interface and captured as an image. Rendered images include the target object, the Barrett hand, normals of the fingers (red lines, not always visible), friction cones in contact locations and coordinate axes. When coordinate axes have not been considered useful they have been removed through an image editing program before being included in this document.

6.2.1 *Picking*

Picking grasps have been optimized according to the approximate affordance function $\tilde{F}_{pick}$ described in Section 4.3.2 and reported in Algorithm 5 with the default hyperparameters of Table 6.2.

Algorithm 5

1: **function** $\tilde{F}_{pick}(E_h)$
2: **return** $-\sum_{i=1}^{6} E_h[i]$
3: **end function**

Picking grasps (Figures 6.7 6.9 6.8) generally wrap around the center of mass as a direct result of the minimization of the total contact forces for holding against gravity.

Some of the produced grasps do not appear intuitive, such as the grasps produced for picking blades in Figure 6.9, because they exploit features of their physical actuator which are very different from a human hand. In this particular case we observe that the hand achieves a stable grasp by pinching the edge of the blade between the joint of some finger. This pinch provides multiple contacts with very different normals that provide much greater stability of the grasp on a low friction material. The emergence of such solution is very unlikely to happen from human evaluation of grasps, as the human intuition is heavily biased by the human hand with much more fingers, much higher tangential and torsional friction and more susceptible to damage than the metal Barrett hand assumed in our experiments.

The change in sampling policy from the first (Figure 6.7) to the second (Figure 6.8) data collection round did not substantially influence the quality of the resulting grasps, while the employed strategy for the second round displays more variance. As the minimal parameterization sampling was indeed biased around the center of mass, such bias resulted in a positive influence for the picking task which takes an objective advantage around this location in terms of the E_h metric. For this reason more extensive search of grasp locations and angles in the second data collection round did not produce any better result, only increasing the variance in the optimal strategy thus reducing, while not eliminating, the occurrence frequency of the joint pinch strategy.

6.2.2 *Cutting*

Cutting grasps have been optimized according to the approximate affordance function $\tilde{F}_{cut}$ described in Section 4.3.2 and reported in Algorithm 6 with the default hyperparameters of Table 6.2.

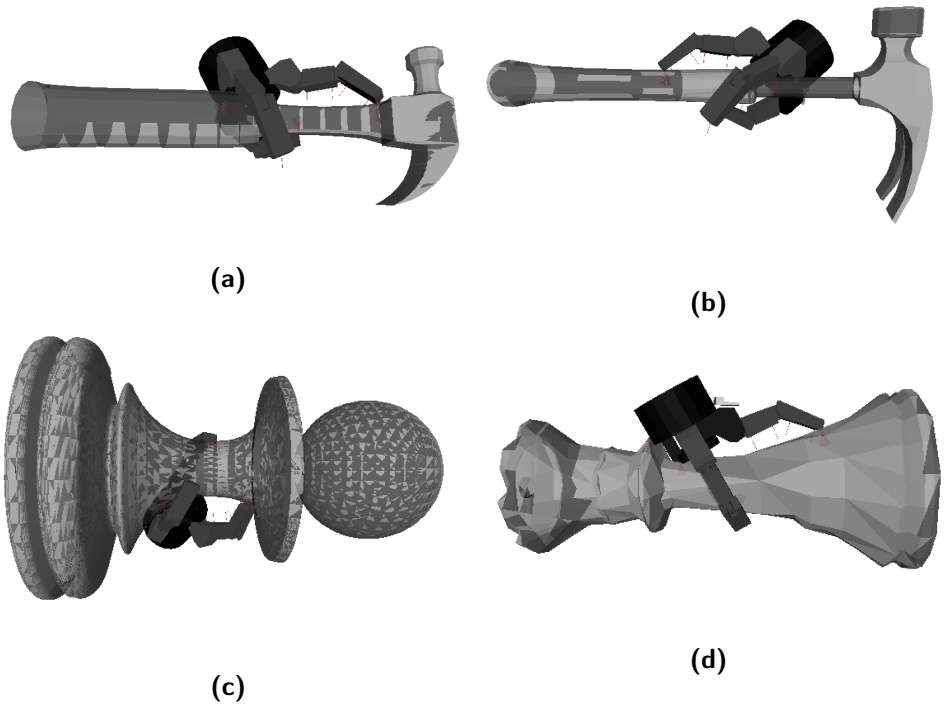


Figure 6.7: Optimized grasps from the picking task on sample objects. First data collection round.

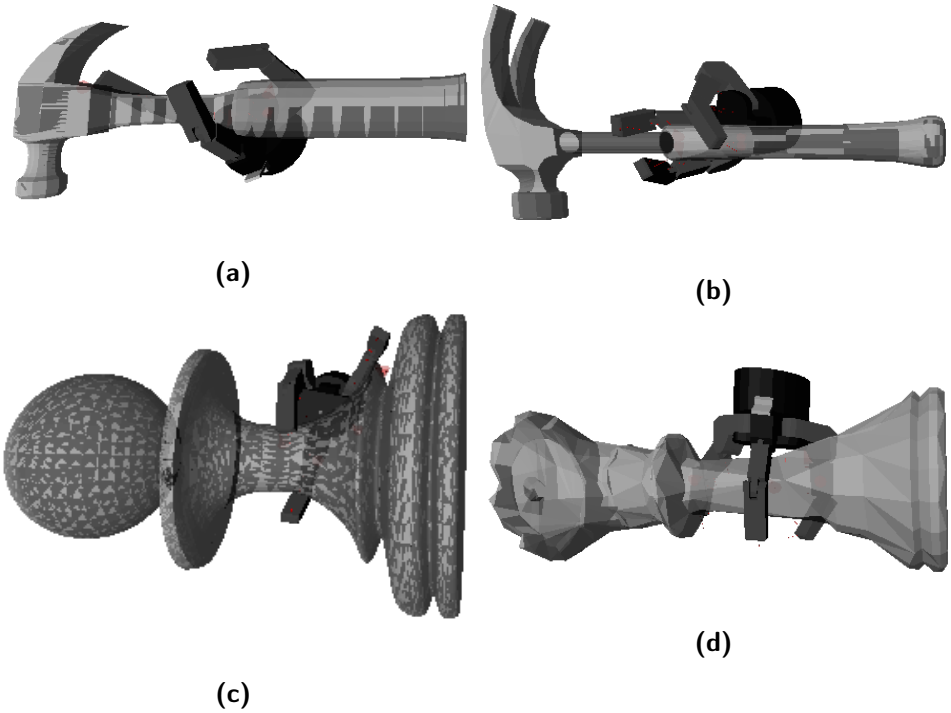


Figure 6.8: Optimized grasps from the picking task on sample objects. Second data collection round.

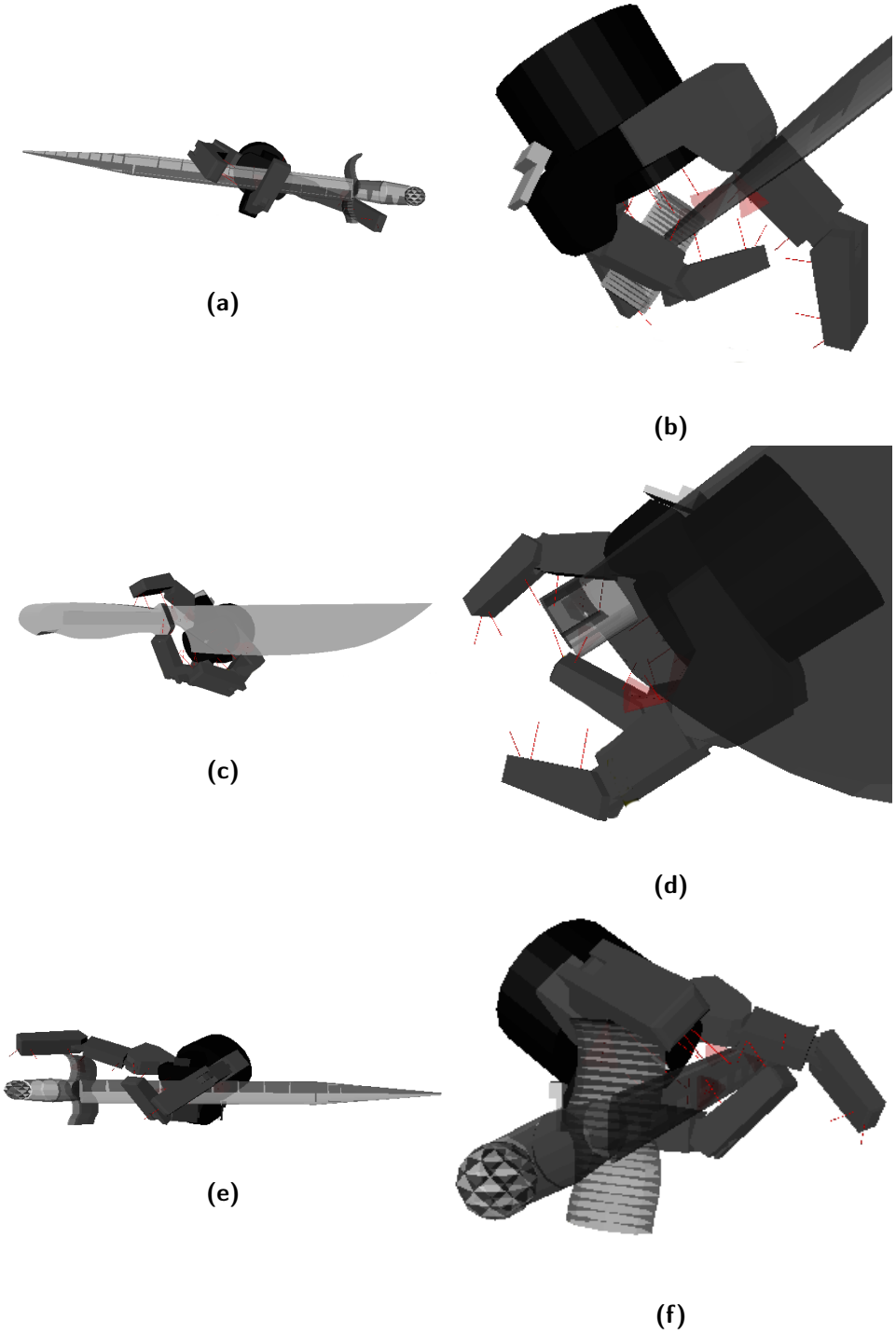


Figure 6.9: Optimized grasps from the picking showing the joint pinch strategy. The edge of the blade is pinched between two joints to improve the stability of the grasp; (b), (d), (f) show the detail of the joint pinch on the blade. (a-b) and (c-d) are from the first data collection round, (e-f) is from the second data collection round.

Algorithm 6

1: **function** $\tilde{F}_{cut}(\epsilon, U_\tau, U_g)$
2: **if** $(\epsilon < \tau_\epsilon \ || \ U_g < \tau_{U_g})$ **then**
3: **return** $-\infty$
4: **else**
5: **return** U_τ
6: **end if**
7: **end function**

On the first data collection round (Figure 6.10) adaptive behaviours are evident: with thin blades the hand exercises pressure on the edge by rotation, using the handle as a fulcrum, while with larger blades where such technique is not feasible a direct pressure is preferred.

On the second data collection round (Figure 6.11) the unbiased parameterization caused the variability of developed strategies to increase substantially. The exploration of grasp and use locations is not biased around the center of mass, thus the selection of the cutting point is much more expressive, as shown in Figure 6.11a where the cut location has been selected on a sharp decoration on the edge of the sword. Moreover, for the same reason optimal grasps move towards the handles as well, being able to obtain more heterogeneous contacts more easily.

However, complete exploration for the use location gave substaintial problems with the inaccurate rendering of blade edges in meshes, as anticipated in Section 5.3.5. Specifically, the edge of the kitchen knife model (Figure 6.4e) displays a very large and flat edge and sharp angles on the handle, attracting cutting locations towards the handle. This behaviour, while correct from a formal point of view, is undesired in our qualitative validation, thus to preserve semantic coherence the grasp shown in Figure 6.11c has been obtained by explicitly excluding all grasps with use location on the handle.

6.2.3 *Beating*

Beating grasps have been optimized according to the approximate affordance function $\tilde{F}_{beat}$ described in Section 4.3.2 and reported in Algorithm 7 with the default hyperparameters of Table 6.2.

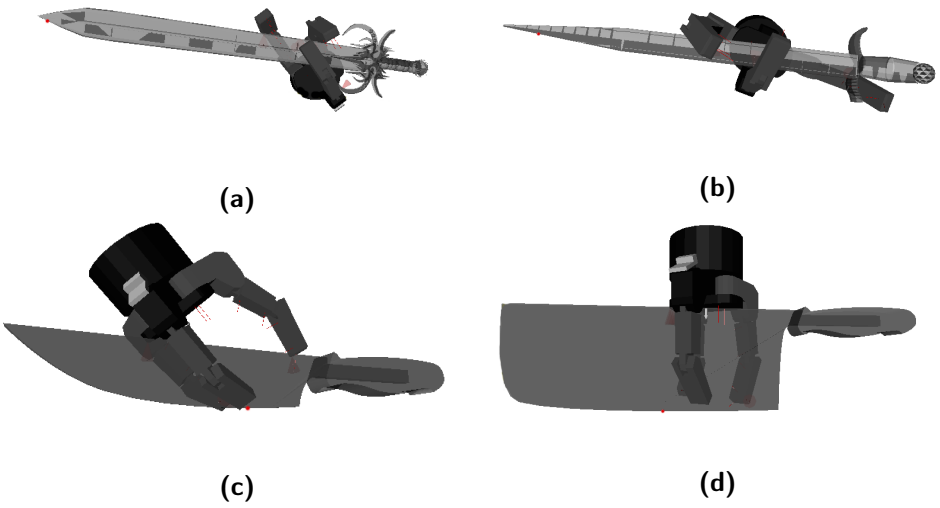


Figure 6.10: Optimized grasps from the cutting task on the first data collection round showing two different cutting strategies: thin blade tools like (a) and (b) exercise pressure by rotation, wide blade tools like (c) and (d) instead prefer a direct pressure strategy.

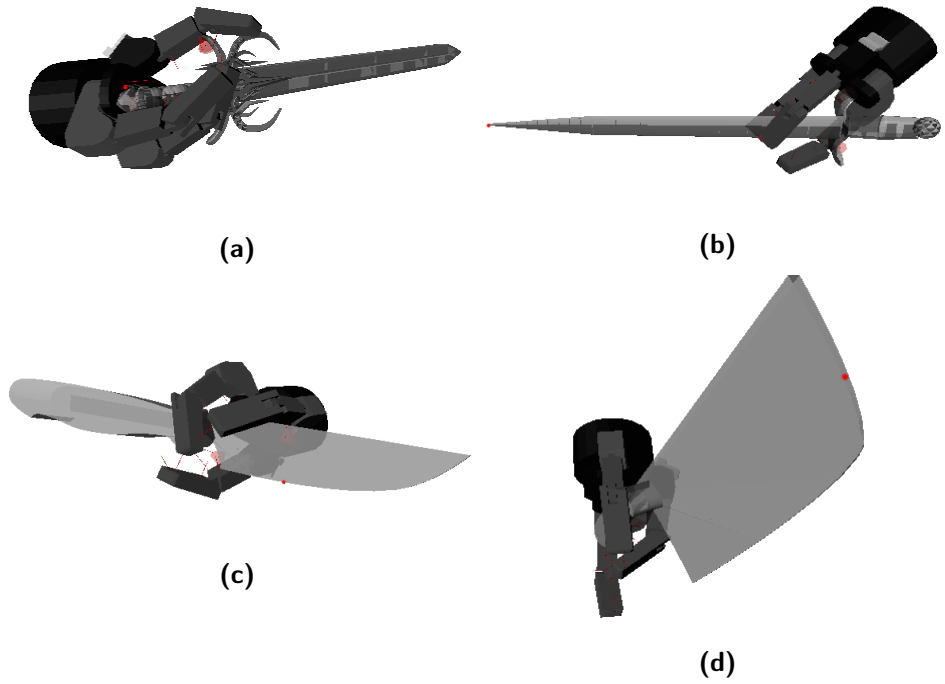


Figure 6.11: Optimized grasps from the cutting task on the second data collection round.

Algorithm 7

function $\tilde{F}_{beat}(\epsilon, \delta, I, E_i)$
  if $(\epsilon < \tau_\epsilon \ || \ \delta < \tau_\delta)$ **then**
    return $-\infty$
  else
    return $\frac{I}{E_i}$
  end if
end function

Beating grasps (Figure 6.12 6.13) display greater variance in general and usually achieve a stable grasp on the object far from the center of mass to increase the rotational inertia of the object and choose a use location very well aligned with the rotation direction to effectively discharge the rotational energy on the target (the values of δ for the optimal grasps are far nearer to 1 than the required values for the selected threshold τ_δ). The use location is selected slightly off the center of mass from the opposite side of the hand to balance the beating impulse and produce a torque that contrasts the rotational inertia of the beating movement.

For what concerns the use of hammers, we must take into consideration that simulated hammers are assumed to be homogeneous in material density, thus their center of mass is effectively on the handle. This important difference with real hammers change the optimal use from beating on their head to beating on their handle.

The second data collection round (Figure 6.13) substantially improved the quality of the beating task results, as more extensive exploration far from the center of mass allows evaluating grasps which are farther from the center of mass and use location. Hammers are now more effectively grasped on the head, which offers more diverse surface for a solid grasp, as the optimal beating point is always on the handle for the same considerations mentioned before. The main strategies for beating with chess pieces did not change, while the easier selection of a farther point of use makes them generally more effective.

6.2.4 *Affordance function parameters fluctuation experiment*

As a further validation test we changed the ϵ threshold requirement for the tasks of beating and cutting according to Table 6.2. The results of such experiment conducted on the model of a typical kitchen knife are shown in Figure 6.14. The produced grasps for beating with a knife display a similar strategy with respect to the ones for more classical objects as seen in Figure 6.12: the grasp is as far as possible from the center of mass, switching to the handle only when greater robustness is required. The cutting task on the

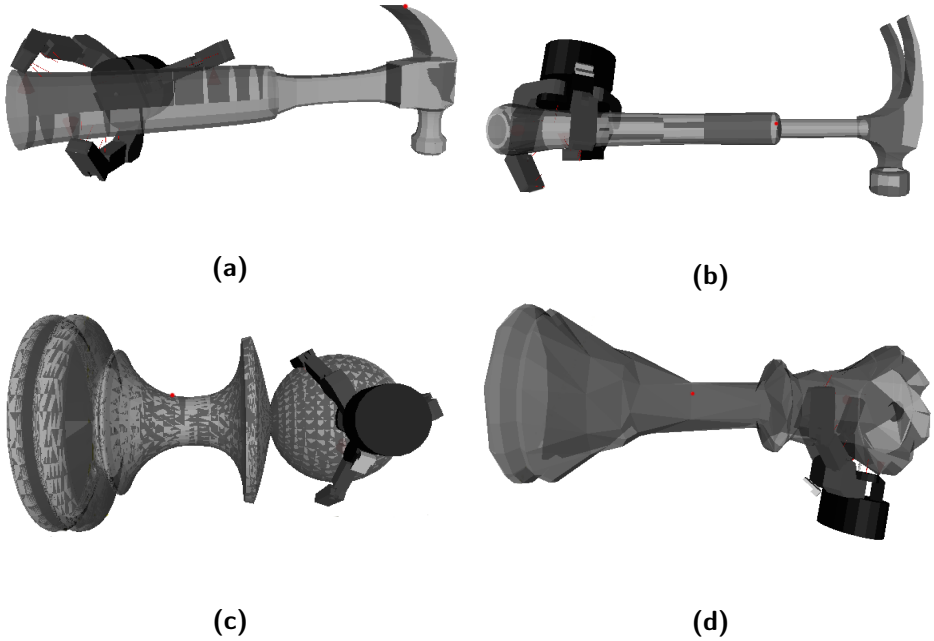


Figure 6.12: Optimized grasps from the beating task on sample objects from the first round of data collection. Notice that the center of mass of hammers in (a) and (b) is on the handle, as the material is assumed homogeneous.

Table 6.2: Affordance function thresholds

Mode name	τ_ϵ	τ_{U_g}	τ_δ
Default	0.3	10	0.95
No robustness required	$-\infty$	10	0.95
Extra robustness required	0.5	10	0.95

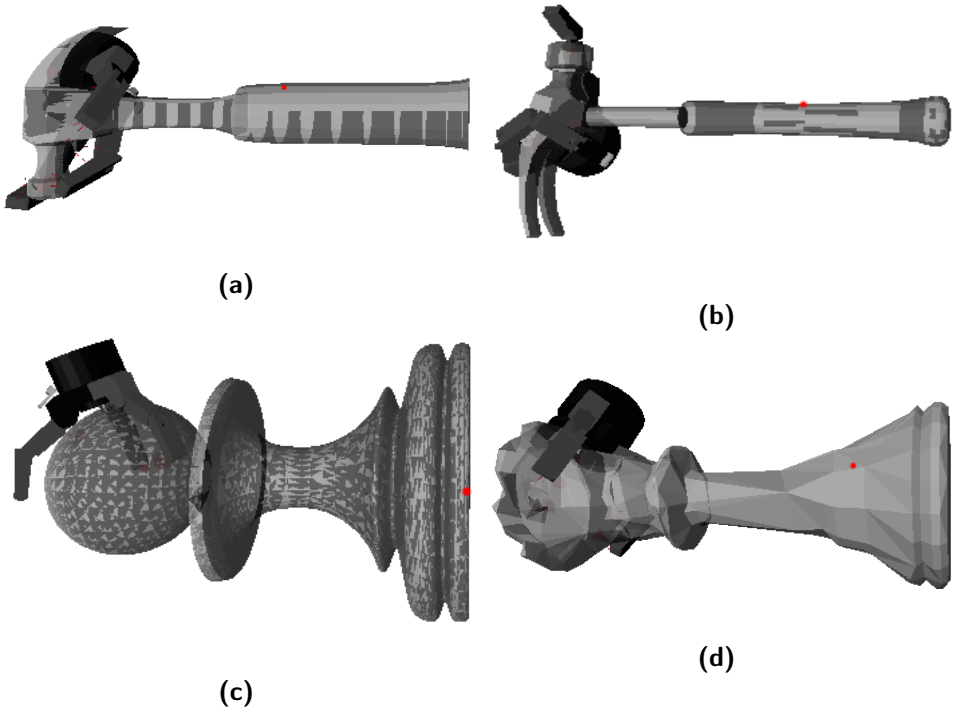
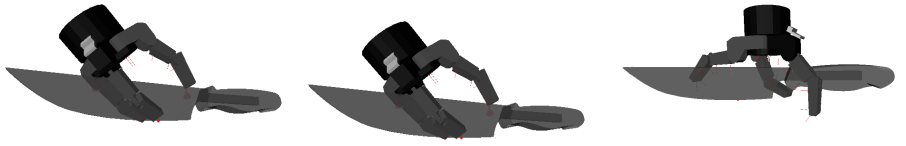


Figure 6.13: Optimized grasps from the beating task on sample objects from the second round of data collection. Notice that the center of mass of hammers in (a) and (b) is on the handle, as the material is assumed homogeneous.

(a) Beating, no robustness required

(b) Beating, default

(c) Beating, extra robustness required



(d) Cutting, no robustness required

(e) Cutting, default

(f) Cutting, extra robustness required

Figure 6.14: Affordance function parameter variability. The grasp strategy varies slightly while changing the requirement of the robustness of grasp.

knife shows less variance, producing robust grasps even when not explicitly required by the fitness function.

6.2.5 *Validation results discussion*

The outcomes of this experiment define the upper bound of performance of the final system as we are selecting the best grasp and use hypothesis directly among all the available data, collected in a perfect information environment.

The three selected tasks produced qualitatively meaningful grasps, sometimes displaying unusual but smart strategies like joint pinching in Figure 6.9, exploiting the features of the robotic hand in use. Most of the grasps produced represent feasible and effective alternatives where the more conventional grasps that a human would perform would fail due to the enormous differences between the human hand, an underlying hypothesis intrinsic in human intuition, and the actual robotic hand which has less fingers, with squared rigid surfaces, low friction coefficient and very limited degrees of freedom.

Nonetheless, the experiment underlined criticalities and limitations on the current system. We have seen that the distribution of data highly influences the outcomes of this validation step, thus the upper bound of the quality of the system, introducing biases in the exploration of alterntives that are difficult to overcome even with millions of samples. Moreover, especially for some tasks like beating, the different materials of each part of an object are an important factor to determine the real best grasp and use hypothesis on the same object, like in beating with hammers (Figure 6.12 and 6.13), which requires further complexity in the simulation and a more informative mesh

dataset than the Princeton Shape Benchmark which is designed for shape matching and classification algorithms.

6.3 LEARNING RESULTS

As a first step for learning, we separated training, validation and test sets based on object meshes. The training set shown in Figure 6.3 counts ten different meshes including many tools, common objects and some uncommon geometries to help generalize to the actual geometry of the mesh, not on the semantic category. The validation set shown in Figure 6.4 counts five different meshes with tools and common objects which highly resemble the shapes found in the training set. Finally, the test set shown in Figure 6.5 counts three objects with a common bottle example (Fig 6.5a), an uncommon tool (Fig 6.5c) which only resembles a single training example, and an axe (Fig 6.5b) whose geometry resembles no other object in the previous sets. Not all objects have been used for learning, as the ones in Figure 6.6 do have either shapes or sizes that are too distant from the training and validation distributions and have been found to be detrimental for our learning task.

6.3.1 *Performance measures*

We evaluate our learned models by benchmarking against a set of metrics that we select to capture the operating performance of the model when plugged into a complete system. Metrics are designed to be reproducible, thus we prefer standard metrics when possible.

CLASSIFIER as the classifier model is intended to be used as a filter, we are mainly interested in the probability of a selected grasp, thus a positive sample, to actually be a true positive (thus we need a high precision score). At the same time we cannot tolerate too many viable grasps to be discarded, thus we have to monitor the recall as well.

Therefore, we draw the precision-recall curves for all classifiers and set a tolerance threshold τ_r on the recall, ranking them by the achieved precision for that value of recall.

REGRESSOR We consider three performance measures for the regressor models:

- **MSE**: the Mean Squared Error (MSE) is the classical metric used to assess basic regression, it gives an overall score of how near the regression goes to real values. As our goal is optimization, not estimation, we consider this metric relevant but incomplete.

- **CA**: the Comparison Accuracy (CA) is obtained by sampling random couples of samples and measuring the standard accuracy in telling which input sample corresponds to a greater value of the metric. This is considered a more specialized indicator than the MSE as optimization within a number of choices is built by comparisons.
- **GMS**: as the regressor model has to find the optimal sample within a set of possible choices, we design the Global Minimum Score (GMS) to quantify directly the optimality that we expect from the chosen sample. Let I_{min} be the input sample that minimizes the predicted output $\mathcal{M}_R^{\Phi}(I_{min})$ over a set $\mathcal{S}$ of samples, then we define the GMS of the model for the set $\mathcal{S}$ as the rate of samples that actually have a greater ground truth value than the ground truth value of I_{min}.

While the MSE and CA performance measures are stable with a high enough number of test samples, we found that the GMS measure as defined above is not. As the GMS is extremely sensible to adversarial examples, which have extremely low score for the learned model and high ground truth value (like, e.g., an object seen from a perspective where it is mostly occluded may look easy to be grasped in the visible center while being in reality very unbalanced, as in Figure 6.19e), the presence of even a single adversarial example within the test set can change drastically the resulting value of GMS.

6.3.1.1 *Stable formulation the Global Minimum Score*

The GMS is our most significant performance measure to discriminate between different models as it quantifies directly the optimality of their candidate best, which is the only sample we preserve while exploring a set of grasp hypotheses. However, the important dependence on the chosen test set of the GMS as it was defined above, even with large test sets independently sampled from the same distribution, makes it a very unreliable performance measure. We want to stabilize the GMS metric while capturing the phenomenon of adversarial samples, which may appear also in real use cases and must not be selectively eliminated. As a first stabilization technique, we repeatedly subsample the test set $\mathcal{S}$ with rate r by extracting a subset $S \in \mathcal{S}$ such that $|S| = \lceil r|\mathcal{S}| \rceil$ and compute the GMS onto S. As we get multiple different values, we can draw a probability distribution and compute its expected value E[GMS]. The results of such technique performed on the trained regressors with $r = \frac{1}{10}$ and 10^5 repetitions over 10^5 test samples are shown in Figure 6.15. The cumulative distribution (up in the figure) of all models rises in significant steps, corresponding to irregular spikes in the density (bottom in the Figure): the expected value of the distribution is still very sensible to the specific adversarial samples that are present within the test set.

In the following we identify the dependence of the final GMS distribution with each sample in the test set to understand the reason of its instability

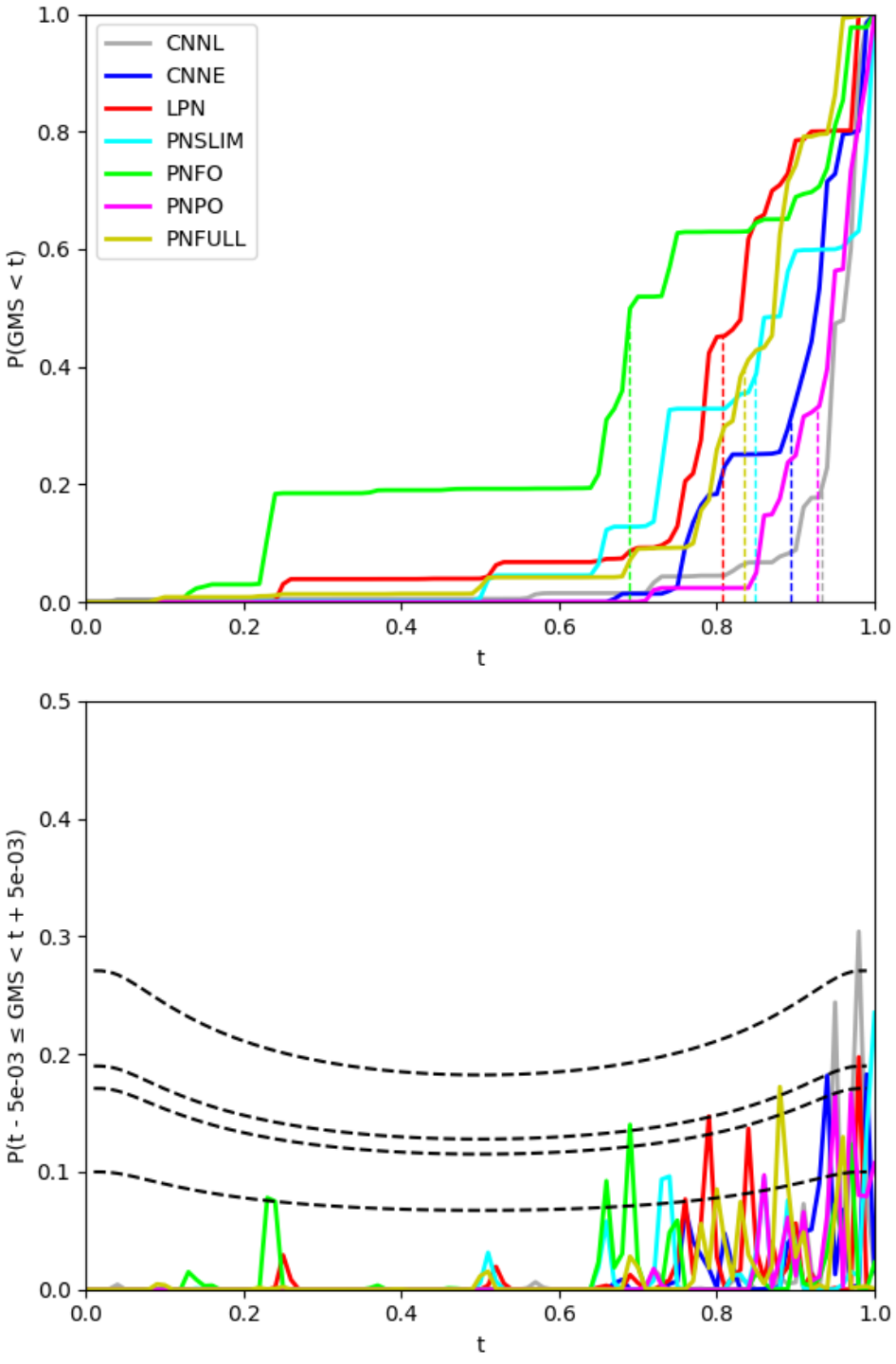


Figure 6.15: Unstable GMS distributions for regression models, with density peak prediction curves.

even with large test sets, showing that undesired spikes in the density are bound to happen with a predictable pattern in height and in different locations for any test set $\mathcal{S}$, being determined by few samples in the test set; then we validate our modeling by predicting the height of some spikes in Figure 6.15.

Let $\alpha \in \mathcal{S}$, let $\phi : \mathcal{S} \mapsto \mathbb{R}$ be the ground truth function and $g_S : S \mapsto \mathbb{R}$ be the relative ground truth score function defined as:

$$g_S(\alpha) = \frac{|\{\beta \in S : \phi(\beta) \geqslant \phi(\alpha)\}|}{|S|} \tag{6.2}$$

Let I_a^b be the open interval between reals a and b, then we can formulate the interval probability of the GMS following our current sampling technique as:

$$\begin{aligned}
&P(GMS \in I_{t-\delta}^{t+\delta}) \\
&= P\left(g_S(\arg\min_{\beta \in S}(\mathcal{M}_R^\Phi(\beta)) \in I_{t-\delta}^{t+\delta}\right) && (6.3)\\
&= P\left(\bigcup_{\alpha \in \mathcal{S}} g_S(\alpha) \in I_{t-\delta}^{t+\delta} \wedge \alpha \in S \wedge \alpha = \arg\min_{\beta \in S}(\mathcal{M}_R^\Phi(\beta))\right) && (6.4)\\
&= P\left(\bigcup_{\alpha \in \mathcal{S}} g_S(\alpha) \in I_{t-\delta}^{t+\delta} \wedge \alpha \in S \wedge \nexists \beta \in S : \mathcal{M}_R^\Phi(\beta) < \mathcal{M}_R^\Phi(\alpha)\right) && (6.5)
\end{aligned}$$

Let us now consider that we are performing the union of perfectly disjoint events as there must be a unique value of α such that $\alpha = \arg\min_{\beta \in S}(\mathcal{M}_R^\Phi(\beta))$, therefore we can sum the probabilities of the single events. Moreover, for $|S| \to \infty$, we can assume that the three events of Equation 6.5 are independent and can be factored as:

$$\begin{aligned}
&P(GMS \in I_{t-\delta}^{t+\delta}) \\
&= \sum_{\alpha \in \mathcal{S}} P(\alpha \in S)\, P\left(g_S(\alpha) \in I_{t-\delta}^{t+\delta}\right) P\left(\nexists \beta \in S : \mathcal{M}_R^\Phi(\beta) < \mathcal{M}_R^\Phi(\alpha)\right) && (6.6)\\
&= \sum_{\alpha \in \mathcal{S}} rP\left(\frac{Bin(|S|, g_S(\alpha))}{|S|} \in I_{t-\delta}^{t+\delta}\right) P\left(\nexists \beta \in S : \mathcal{M}_R^\Phi(\beta) < \mathcal{M}_R^\Phi(\alpha)\right) && (6.7)
\end{aligned}$$

where we substituted $P(\alpha \in S) = \frac{|S|}{|\mathcal{S}|}$ with the sampling rate r and we assumed independent sampling for $g_S(\alpha)$, that therefore is distributed as a normalized binomial $\frac{Bin(|S|, g_S(\alpha))}{|S|}$, with mean $g_S(\alpha)$ and variance $\frac{g_S(\alpha)(1-g_S(\alpha))}{|S|}$.

We can observe that the model $\mathcal{M}_R^\Phi$ defines a strict ordering among samples $\alpha \in \mathcal{S}$, therefore we can index samples $\alpha_0, \alpha_1, \ldots \alpha_{|\mathcal{S}|-1}$ in inceasing

order of $\mathcal{M}_R^\Phi(\alpha)$. We can rewrite the righmost probability in Equation 6.7 for a specific α_k as:

$$\begin{aligned} &P\left(\nexists \beta \in S : \mathcal{M}_R^\Phi(\beta) < \mathcal{M}_R^\Phi(\alpha_k)\right) \\ &= P\left(\bigwedge_{j<k} \alpha_j \notin S\right) \qquad (6.8) \\ &= \prod_{j<k} P(\alpha_j \notin S) = (1-r)^k \qquad (6.9) \end{aligned}$$

Therefore we can obtain a final formulation for the discrete density of the GMS for $|\mathcal{S}| \to \infty$ (Equation 6.10) or a continuous approximation based on the observation that Binomials are sums of iid Bernoulli distributions (Equation 6.11):

$$P(\text{GMS} \in I_{t-\delta}^{t+\delta}) = \sum_{k=0}^{|\mathcal{S}|-1} \left(r(1-r)^k P\left(\frac{\text{Bin}(\lceil r|\mathcal{S}| \rceil, g_{\mathcal{S}}(\alpha_k))}{\lceil r|\mathcal{S}| \rceil} \in I_{t-\delta}^{t+\delta} \right) \right) \qquad (6.10)$$

$$p_{\text{GMS}}(t) = \sum_{k=0}^{|\mathcal{S}|-1} \left(r(1-r)^k \mathcal{N}\left(g_{\mathcal{S}}(\alpha_k), \frac{g_{\mathcal{S}}(\alpha_k)(1-g_{\mathcal{S}}(\alpha_k))}{\lceil r|\mathcal{S}| \rceil} \right) \right) \qquad (6.11)$$

where, with a slight abuse of notation, by $\mathcal{N}(\mu, \sigma^2)$ we denote the usual density function of the normal distribution with mean μ and variance σ^2. Equation 6.11 shows that the density function p_{GMS} is well approximated by a mixture of gaussians whith exponentially decaying contributions of each sample of the test set, and where each gaussian has extremely low variance. This produces a peak pattern in the discrete density, where each peak k is located around $t = g_{\mathcal{S}}(\alpha_k)$ with height $r(1-r)^k$ when the sampling interval is wide enough to capture most of the density of the corresponding binomial around its mean, while for larger sampling intervals some peaks can overlap and be summed. In Figure 6.15 we verify this model experimentally by drawing the curves for the expected overlapped peak height for samples $\{\alpha_0\}, \{\alpha_1, \alpha_2\}, \{\alpha_0, \alpha_1\}, \{\alpha_0, \alpha_1, \alpha_2\}$; the curvature is due to the gaussian component integrated over the sampling interval, which has a different variance based on the location of the peak.

The variability of the resulting GMS density even with very large test sets is due to the fact that its density function depends mainly on few peculiar samples of the test set, which is rooted to the intrinsic instability of the $\min$ function over ordered sets. As we cannot drop the use of the $\min$ function, we act upon the Equations 6.10 and 6.11 to spread the importance evenly to all samples. We achieve this by nearing the exponential decay coefficient $1-r$ to 1, thus for $r \to 0$, which in turn would drastically reduce the size

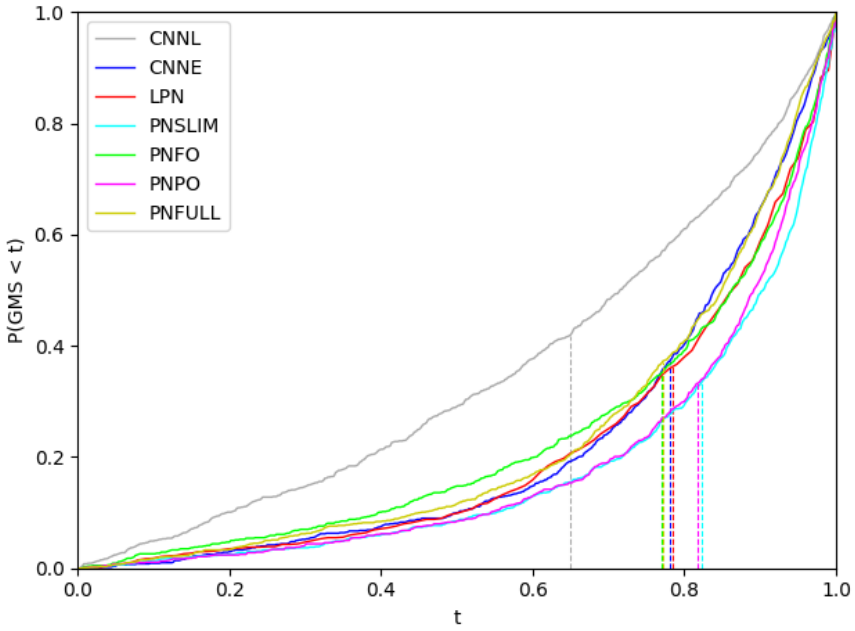


Figure 6.16: GMS distributions for regression models.

of subsets S for a fixed cardinality of the whole test set $\mathcal{S}$. We bring this reasoning to the limit by dropping the sampling phase in favour of partitioning of $\mathcal{S}$: each data sample is used only once and compared against a partition of the original test set. The new method is much more stable, producing the distributions in Figure 6.16, therefore it has been used for the evaluation of regressor models.

6.3.2 *Hyperparameters*

We test many variations of Convolutional Neural Network (CNN) and PointNet (PN) models varying many hyperparameters. All models in Tables 6.3 and 6.4 have been trained on both the classification and regression tasks; in the following we explain the meaning of each hyperparameter and name used in defining our models.

Names and hyperparameters used for both CNN and PointNet models:

- **Model code**: a conventional code used to identify a model defined by a set of hyperparameters. It summarizes the main features of the model.
- λ_0: the initial learning rate at the starting epoch. We refer to Section 5.5.2 for the decaying update rule.

Table 6.3: CNN Architectures

Model code	λ_0	Dropout	Early/Late	$\mathcal{H}$	$\mathcal{C}$	Params
CNNL	10^{-4}	0.75	L	192	36	21070K
CNNE	10^{-4}	0.75	E	192	36	21070K

Table 6.4: PointNet Architectures

Model code	Reduction method	λ_0	Dropout	$\mathcal{T}_C$	$\mathcal{T}_F$	Params
PNFULL	Max pool	10^{-4}	0.5	yes	yes	3471K
PNFO	Max pool	10^{-4}	0.5	no	yes	2667K
PNPO	Max pool	10^{-4}	0.5	yes	no	1613K
PNSLIM	Max pool	10^{-4}	0.5	no	no	810K
LPN	Xception [53]	10^{-4}	0.75	yes	yes	22019K

- **Dropout**: the dropout keep rate used in training the network. It is the probability that an activation is not zeroed into a dropout layer.
- **Params**: total count of trainable parameters. This is a dependent parameter that we include to give an overall measure of the complexity of the model.

Names and hyperparameters used for CNN models:

- **Early/Late**: whether early fusion or late fusion is used in the CNN, as explained in Section 6.1.1.
- $\mathcal{H}$: number of hidden units used on the top of the CNN, after convolutions.
- $\mathcal{C}$: number of convolutional layers in the encoder section of the CNN.

Names and hyperparameters used for PN models:

- **Reduction method**: the method employed to reduce the point-wise features extracted by the PointNet. As anticipated in Section 6.1.3 we propose a novel reduction method for the PointNet architecture to capture local features, constituted by an Xception [53] model instead of the classical max pooling layer.
- $\mathcal{T}_C$: whether the coordinate transformer net is included into the model.
- $\mathcal{T}_F$: whether the feature transformer net is included into the model.

Table 6.5: Classifiers fixed recall precision benchmark ranking

Model code	Cross entropy	Precision
LPN	**0.3842**	**0.856**
PNFULL	0.4398	0.829
PNFO	0.4403	0.820
PNPO	0.4628	0.819
CNNE	0.4400	0.818
PNSLIM	0.4696	0.800
CNNL	0.5537	0.741

6.3.3 *Classifier*

The classifier $\mathcal{M}_C^\Phi$ filters viable grasps from input data based on the pregrasp and on the input range image with camera-in-hand perspective. All architectures in Tables 6.3 and 6.4 have been trained on this task using 500K training samples from the second round of data collection.

In Figure 6.17 we plot the precision-recall curves of all tested classifiers, obtained by varying the acceptance thresholds for the output of the classifier. We rank the available models by the precision that they can achieve with a fixed recall level of 0.75, drawing the ranking in Table 6.5.

While most models have very similar performances, the Local PointNet results as the clearly superior model for the classification task, both from a qualitative evaluation of the precision-recall curve and from the quantitative ranking. On the other side, the late fusion Xception architecture performed the worst, with far greater loss than its early fusion counterpart, suggesting that the geometrical features that are relevant for the classification of suitable grasps are significantly dependent on the pregrasp p_0.

6.3.4 *Regressor*

The regressor $\mathcal{M}_R^\Phi$ infers the metric vector ϕ for later computation of the affordance function relative to the input sample. In this thesis we trained regressor networks to infer only the value of $\sum_{i=1}^{6} E_h[i]$ for testing with the picking affordance function on the picking task. Output values are linearly normalized in the interval $[0, 1]$ from the original domain $[0, \tau_{E_h}]$ granted by the assumption that input values come from the positive class of viable grasps. All models in Tables 6.3 and 6.4 have been trained on the same regression task using 500K training samples from the second round of data collection.

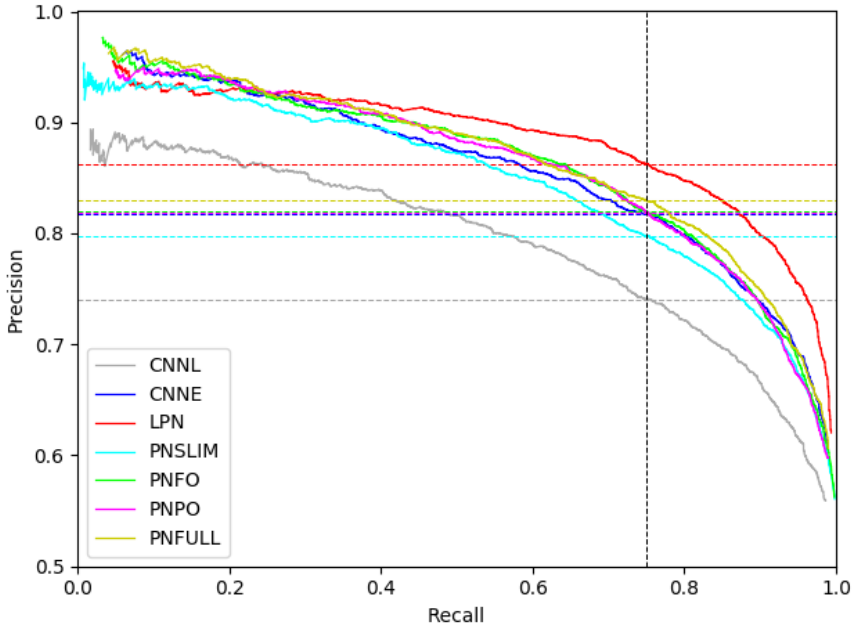


Figure 6.17: Precision-recall curves for classification models.

Table 6.6: Regression results

Model code	MSE	CA	E[GMS]
PNPO	**0.033**	**0.73**	0.818
PNSLIM	0.034	0.72	**0.824**
PNFULL	0.034	0.71	0.771
PNFO	0.035	0.72	0.770
LPN	0.037	0.69	0.785
CNNE	0.038	0.69	0.781
CNNL	0.050	0.58	0.649

Results on the regression task are shown in Table 6.6, and GMS distributions computed as described earlier with 500K test samples are plotted for comparison in Figure 6.16.

All models appear to have a similarly shaped distribution of GMS, while most of the models locate at $E[GMS] \approx 0.775$. Similarly to the case of the classification task, the late fusion Xception model performs much worse than the others, including the early fusion Xception that stays in the main cluster. Surprisingly enough, the two lightest models perform significantly better in terms of all considered metrics, while the distinction between the two is not significant.

6.4 OPTIMIZATION OVER LEARNED MODELS

As a final validation step we test the learned system as a whole: based on the benchmarks of all available models, we choose a specific implementation for $\mathcal{M}_C^\Phi$ and $\mathcal{M}_R^\Phi$ with a pass threshold to accept samples based on the precision-recall curve of the chosen $\mathcal{M}_C^\Phi$.

Referring to Algorithm 1 described in Section 4.2.1, we select $\mathcal{M}_R^\Phi$ as the PointNet regressor with point transform only (PNPO), model $\mathcal{M}_C^\Phi$ as the Local PointNet (LPN) with threshold $\tau_C = 0.6355$ that corresponds to the value of 0.75 recall on the test set benchmark, and fail value $v_{fail} = 1.0$ which is the maximum theoretical output of the regressor model.

Our model $\mathcal{M}^\Phi$ predicts a normalized value in $[0, 1]$ for the sum of the hand effort on hold feature $\sum_{i=1}^{6} E_h[i]$ for viable grasps and 1.0 for not viable grasps, therefore we can search for a grasp that minimizes the output of $\mathcal{M}^\Phi$ to equivalently search for one that maximizes the affordance function $\tilde{F}_{pick}$ described in Algorithm 5. Therefore we produce an algorithm to extract the best grasp hypothesis for picking from vision by combining our evaluation model $\mathcal{M}^\Phi$ with a minimization algorithm. For the selection of the minimization algorithm we tested on standard smart optimization algorithms available in the scikit-optimize package [55] that try to build a simple model of the objective function, assumed expensive, while evaluating samples of it to make smart guesses on where the minimum is expected to lie, performing at most n_{tries} function evaluations. We evaluated the efficacy of a method based on the rate of acceptance of its guesses through the classifier, while looking for a trade-off between efficacy and run time. For our experiment we choose the optimizer based on the random forest regressor as it could reach very effective rates of acceptance while still maintaining an acceptable run time and variance between proposed samples.

6.4.1 *Grasp extraction*

The chosen optimizer models the objective function it is required to optimize on a domain as simple as an hypercube from which it draws smart guesses on the next try to find the minimum. Our domain of pregrasps $\mathcal{P}$ is definitely non-standard, thus we need to produce a biunivocal mapping from an hypercube $[0,1]^n$ of dimension n to $\mathcal{P}$. We want to achieve the following properties on this mapping:

1. Preserve **uniformity**: under the assumption of uniform distribution of input parameters, we require that the image through the designed mapping follows a uniform distribution in the target space. Note that the uniformity of our target space of pregrasps $\mathcal{P}$ must be defined according with the task of exhaustively searching grasp hypotheses. This property is fundamental to avoid the introduction of search biases into the model, which could result to detrimental effects as with the bias present in the first round of data collection as discussed in Section 6.2.

2. Be **deterministic**: as we want to obtain exactly reproducible results, repeatability of optimization runs is an extremely positive property, therefore we wish to introduce no random component within the process of computing the mapping

We determine the dimensionality of the space $\mathcal{P}$ by decomposing pregrasps in a grasp orientation γ, a target location λ on the object surface, and the DoF δ of the hand:

- The grasp orientation γ is itself a rotation in $\mathbb{R}^3$, which can be uniquely described by three Euler angles, which are a minimal encoding, thus it has dimension three.

- The target location λ is itself a point on a bidimensional surface in space, thus it has two dimensions.

- The DoF vector δ highly depends on the hand used and the eigengrasps that we plan to use to reduce its dimensionality. While it could easily surpass the dimensionality of 20 (for the case of the human hand), in our case we could reduce it to dimensionality one, as discussed in Section 5.3.1.

.

The minimal overall target dimensionality is six, thus we look for a mapping $[0,1]^6 \mapsto \mathcal{P} \subset H \times S(O) \times [0,1]$ where H is the set of all unit quaternions, that encode rotations in $\mathcal{R}^3$, and $S(O)$ is the set of points on the surface of a fixed object O. Moreover, we define the uniformity of the target pregrasp

space $\mathcal{P}$ as being uniform into the respective subspace of H, $S(O)$ and $[0,1]$ that are included in $\mathcal{P}$.

Given an input sample $x \in [0,1]^6$, we compute the corresponding unique pregrasp in the following steps:

1. As the whole domain $[0,1]$ of the pregrasp DoF is fully available independently on the other dimensions, it can be determined first. It is trivially obtained by identity with the first component of x.

2. We want to consider only grasp approach directions coming from outside of the object, therefore the legal pregrasp quaternion subspace of H depends on the particular grasp target location chosen, which must be selected first. We describe in Section 6.4.1.1 the selection of the grasp target location from the second and third components of the input sample x uniformly with respect to the surface of the object mesh.

3. Finally we compute the quaternion encoding the pregrasp orientation such that the approach direction is towards the object and within a cone around the normal to the grasp target location. The exact procedure for this computation of a parameterized uniformly distributed quaternion within a constrained subspace of H is explained in Section 6.4.1.2.

6.4.1.1 *Uniform location in mesh*

We decompose again the problem of determining a location on the mesh uniformly with respect to the surface of the mesh from a generator $(s_0, s_1) \in [0,1]^2$ into two subproblems: selecting a target triangle on the mesh and selecting a specific point on the triangle.

TRIANGLE SELECTION Given a set of mesh triangles $\mathcal{T} = \{t_1, t_2, \dots t_n\}$ with areas $a(t_1), a(t_2), \dots a(t_n)$ we want to select a triangle $\bar{t}$ from input s_0 such that:

$$s_0 \sim U(0,1) \implies P(\bar{t} = t_i) = \frac{a(t_i)}{A}, \tag{6.12}$$

where we take $A = \sum_{j=1}^{n} a(t_j)$ to be the total surface area of the mesh. As the target distribution of $\bar{t}$ follows a categorical over the n triangles, we partition the $[0,1]$ domain interval of s_0 in n subsequent intervals $I_1 \dots I_n$ such that the length of interval I_i is exactly $\frac{a(t_i)}{A}$. In particular, if we name $A_i = \sum_{j=1}^{i} a(t_j)$ with $A_0 = 0$, we define the succession of intervals as $I_i = [\frac{A_{i-1}}{A}, \frac{A_i}{A})$, and select the triangle:

$$\bar{t} = t_i \in \mathcal{T} : s_0 \in I_i \tag{6.13}$$

Thus satisfying the condition in Equation 6.12 while defining a deterministic mapping $[0,1] \mapsto \mathcal{T}$. This search is implemented as a binary interval search over the array of precomputed values $A_0, A_1, \dots A_n$.

POINT SELECTION ON TRIANGLE After the selection of a target triangle $\bar{t} = t_{\bar{i}}$ at index $\bar{i}$, the first problem is understanding the hypothesis to be made on the distribution of the input samples s_0 and s_1. While s_1 is trivially a uniform distribution in the unit interval, s_0 is now conditioned by the selection of the triangle $t_{\bar{i}}$, thus the hypothesis on its distribution is changed.

Let us observe that, if $u \sim U(a_1, b_1)$, and $a_1 \leqslant a_2 \leqslant a_3 \leqslant b_3 \leqslant b_2 \leqslant b_1$ then:

$$\begin{aligned} P(u \in (a_3, b_3) | u \in (a_2, b_2)) &= \frac{P(u \in (a_3, b_3) \cap (a_2, b_2))}{P(u \in (a_2, b_2))} \\ = \frac{P(u \in (a_3, b_3))}{P(u \in (a_2, b_2)} &= \frac{\frac{b_3 - a_3}{b_1 - a_1}}{\frac{b_2 - a_2}{b_1 - a_1}} = \frac{b_3 - a_3}{b_2 - a_2} \end{aligned} \tag{6.14}$$

therefore the distribution of a uniform conditioned to the belonging to a subinterval is a uniform over that same subinterval. The same applies to our input sample s_0 which is conditioned to belonging to $[\frac{A_{\bar{i}-1}}{A}, \frac{A_{\bar{i}}}{A})$, therefore we update it to s'_0 to restore the distribution to uniform to the unit interval:

$$s'_0 = \frac{s_0 - \frac{A_{\bar{i}-1}}{A}}{\frac{A_{\bar{i}}}{A} - \frac{A_{\bar{i}-1}}{A}} = \frac{A s_0 - A_{\bar{i}-1}}{A_{\bar{i}} - A_{\bar{i}-1}} \tag{6.15}$$

We now extract a point P on the triangle $\bar{t}$ with vertices v_1, v_2, v_3 uniformly with respect to its surface by the two input samples s'_0, s_1 which are assumed to be independent and uniform in the unit interval.

Referring to Figure 6.18, we first determine by the coefficient $r_1 \in [0,1]$ the segment $\overline{v'_2 v'_3}$ parallel to $\overline{v_2 v_3}$, and then the exact point P within segment $\overline{v'_2 v'_3}$ by the coefficient $r_2 \in [0,1]$. This is translated in the formula:

$$\begin{aligned} P &= r_2 v'_2 + (1 - r_2) v'_3 \\ &= r_2((1 - r_1)v_1 + r_1 v_2) + (1 - r_2)((1 - r_1)v_1 + r_1 v_3) \\ &= (1 - r_1)v_1 + r_1 r_2 v_2 + r_1(1 - r_2)v_3 \end{aligned} \tag{6.16}$$

While r_1 must select the segment with a probability density proportional to its length, the final selection of P by r_2 is trivially uniform, thus we can directly derive $r_2 = s_1$. We set $r_1 = f(s'_0)$ for some function $f : [0,1] \mapsto [0,1]$ with the following condition on the density function of r_1:

$$p_{r_1}(x) = \frac{|\overline{v'_2 v'_3}|}{a(\bar{t})} \propto |\overline{v'_2 v'_3}| \propto x \tag{6.17}$$

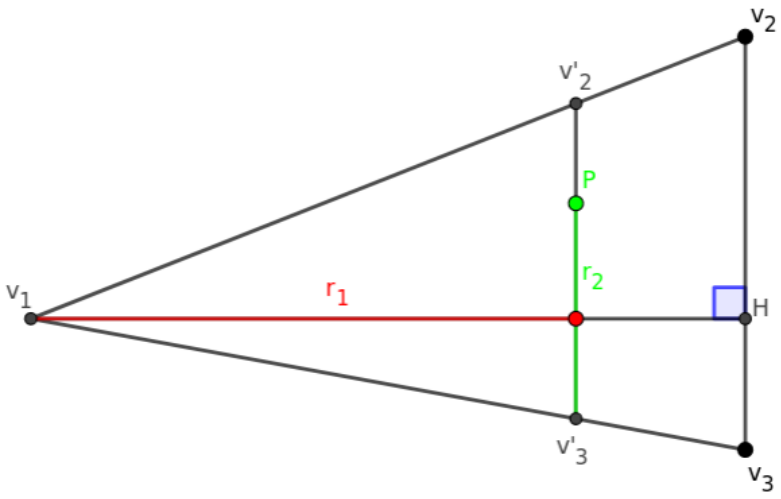


Figure 6.18: Triangle uniform point selection.

Thus, as we have $p_{s'_0}(x) \propto 1$ being s'_0 uniform, we have the following condition on f:

$$\begin{aligned} \frac{\partial f^{-1}(x)}{\partial x} &\propto x \\ f^{-1}(x) &\propto x^2 \\ f(x) &\propto \sqrt{x} \end{aligned} \tag{6.18}$$

The constant for $f(x)$ is determined to fit the image space to $[0, 1]$, thus we set $r_1 = \sqrt{s'_0}$ and we obtain the final formula for the extracted point P:

$$P = \left(1 - \sqrt{s'_0}\right) v_1 + \sqrt{s'_0} s_1 v_2 + \sqrt{s'_0}\,(1 - s_1)\, v_3 \tag{6.19}$$

6.4.1.2 *Uniform constrained quaternion*

The selection of a uniform constrained quaternion intrinsically depends on the selection of the target triangle $\bar{t}$ and in particular on the direction of its normal $\bar{n}$ that is the axis of the cone of legal approach directions, according to Equation 5.2.

We first encode this dependency into a rotation $R_{\bar{n}}$ that maps the z versor $v_z = [0, 0, 1]$, corresponding to the grasp approach direction, into $\bar{n}$, so that we only need to extract a rotation R_{τ_g} that respects the property of Equation 5.2 only around the z axis, and the desired rotation is the composition of the two. Rotation $R_{\bar{n}}$ is trivially obtained as a reference change matrix where the image of v_z is $\bar{n}$ and the images of v_x and v_y are any two versors that are normal to each other and to $\bar{n}$.

This step normalizes the legal regions $\mathcal{L}(\bar{t})$ of accepted quaternions according to Equation 5.2 into a standard legal region $\mathcal{L}'$ which does not depend on $\bar{t}$ and follows the condition:

$$\mathcal{L}' = \{h \in H : R_h(v_z) \cdot v_z \geqslant \tau_g\}, \tag{6.20}$$

where R_h is the rotation encoded by quaternion h.

Our method to determine R_{τ_g} is based on the method from Shoemake et al. [4] in analogy to our unbiased grasp parameterization method described in Section 5.3.1. The extra requirement in this case is that we already have the uniform samples, which cannot be sampled again, thus the re-sampling method used before is not viable anymore.

According to [4], given three values $r_1, r_2, r_3 \sim U([0,1])$ we can sample a random quaternion $h = S(r1, r2, r3)$ uniformly within the space of rotations with the simple expression, which we call S:

$$h = \left(\sqrt{1-r_1}\sin 2\pi r_2, \sqrt{r_1}\cos 2\pi r_3, \sqrt{r_1}\sin 2\pi r_3, \sqrt{1-r_1}\cos 2\pi r_2\right), \tag{6.21}$$

where we swapped the x and z components from the original formula. This modification does not change the properties of the distribution as it only mirrors the rotation axis of each quaternion with respect to the plane $x = z$.

Our objective is to obtain a procedure S' that behaves like the resampling method from Section 5.3.1 without the need of resampling, thus respecting the following condition:

$$\begin{aligned} &r_1, r_2, r_3, r'_1, r'_2, r'_3 \sim U([0,1]) \text{ iid} \\ &h' = S'(r'_1, r'_2, r'_3) \quad h = S(r_1, r_2, r_3) \\ &\forall H' \subseteq H \quad P(h' \in H') = P(h \in H' | h \in \mathcal{L}') \end{aligned} \tag{6.22}$$

We recall that a quaternion $h = (h_w, h_x, h_y, h_z)$ encodes the rotation matrix:

$$R_h = \begin{bmatrix} 1-h_y^2-h_z^2 & 2h_xh_y-2h_zh_w & 2h_xh_z+2h_yh_w \\ 2h_xh_y+2h_zh_w & 1-2h_x^2-2h_z^2 & 2h_yh_z-2h_xh_w \\ 2h_xh_z-2h_yh_w & 2h_yh_z+2h_xh_w & 1-2h_x^2-2h_y^2 \end{bmatrix}, \tag{6.23}$$

thus we can compute the image of the z versor v_z according to 6.21 and 6.23:

$$R_h(v_z) = \begin{bmatrix} 2h_xh_z+2h_yh_w \\ 2h_yh_z-2h_xh_w \\ 1-2h_x^2-2h_y^2 \end{bmatrix} = \begin{bmatrix} 2\sqrt{r1(1-r1)}\cos 2\pi(r_3-r_2) \\ 2\sqrt{r1(1-r1)}\,sen\,2\pi(r_3-r_2) \\ 1-2r_1 \end{bmatrix}, \tag{6.24}$$

We can therefore constrain the distribution of sampled quaternions by constraining the z component of the images of v_z, in particular we can enforce the condition in Equation 5.2 by constraining $\tau_g \leqslant 1 - 2r_1 \leqslant 1$, which is equivalent to:

$$0 \leqslant r_1 \leqslant \frac{1-\tau_g}{2} \tag{6.25}$$

Therefore, we can set our procedure $\mathcal{S}'$ as a function of our input uniform parameters r_1, r_2, r_3:

$$\mathcal{S}'(r_1, r_2, r_3) = \mathcal{S}\left(r_1 \frac{1-\tau_g}{2}, r_2, r_3\right) \tag{6.26}$$

In the following, we prove that such expression respects Condition 6.22. We first notice that $r_1 \frac{1-\tau_g}{2} \sim \mathcal{U}([0, \frac{1-\tau_g}{2}])$ and therefore by Equation 6.14, referring to Condition 6.22 we can rewrite:

$$P(h' \in H') = P\left(h \in H' | 0 \leqslant r_1 \leqslant \frac{1-\tau_g}{2}\right) \tag{6.27}$$

Therefore to prove Condition 6.22 we need to prove the equivalence between events $0 \leqslant r_1 \leqslant \frac{1-\tau_g}{2}$ and $h \in \mathcal{L}'$, which is trivially given by the following sequence of equivalent statements:

$$\begin{aligned}
&0 \leqslant r_1 \leqslant \frac{1-\tau_g}{2} \\
&\tau_g \leqslant R_h(v_z)_z \leqslant 1 \\
&\tau_g \leqslant R_h(v_z) \cdot \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^T \leqslant 1 \\
&R_h(v_z) \cdot v_z \geqslant \tau_g \\
&h \in \mathcal{L}'
\end{aligned}$$

Finally, we can determine the final uniform rotation $\mathcal{R}$ from three uniform parameters s_0, s_1, s_2:

$$\mathcal{R}(s_0, s_1, s_2) = R_{\bar{n}} \circ \mathcal{S}\left(s_0 \frac{1-\tau_g}{2}, s_1, s_2\right) \tag{6.28}$$

6.4.2 *Optimization results*

The complete model M^{Φ} as described above has been tested on the objects on the test set and on some objects of the unused set, which we consider valid test objects which are especially difficult. Figures 6.19 and 6.20 show sample grasp results obtained with the random forest optimizer with

$n_{tries} = 1000$ finally reaching on all objects an acceptance rate from the classifier of 0.7 to 0.8 depending on the object.

The resulting grasps resemble the grasps obtained directly from extensive search over all the dataset, shown in Figures 6.7 and 6.8, generally implementing the strategy of wrapping the center of mass. There are failure cases in which the vision from one very partial perspective, with no global information on the object grasps an object far from the center of mass, like in Figures 6.19e and 6.19f where the head of the chess pawn hid completely the geometry of the whole object. In general we observe that simpler objects usually converge fast to similar optimal grasps, while more complex objects like the axe and the chess pawn still display a considerable variability of results even with the current level of n_{tries}. Moreover, as shown in Figure 6.20, the learned models are able to recognize and perform the same strategy of joint pinching that we recognized analysing directly the data as in Figure 6.9.

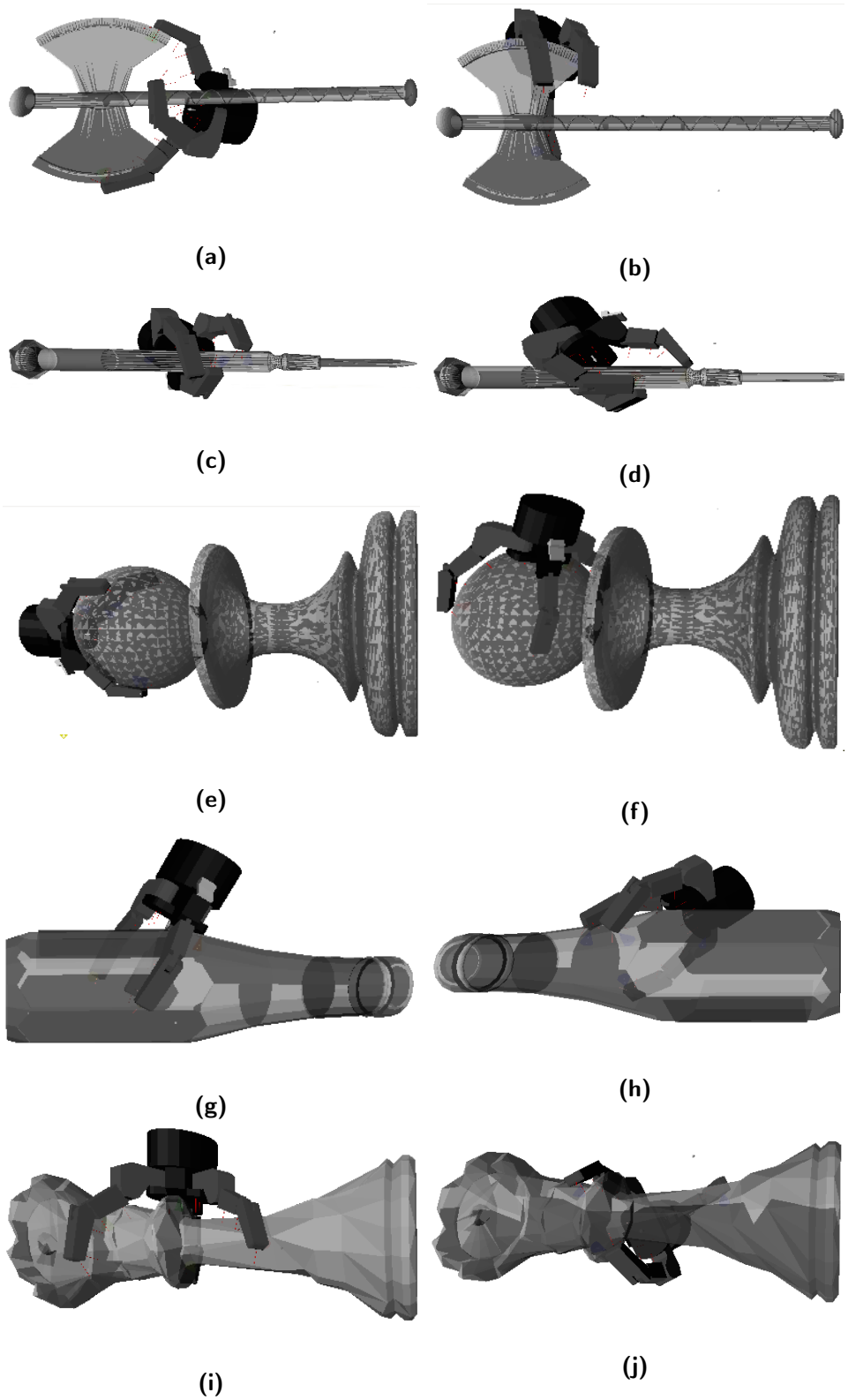
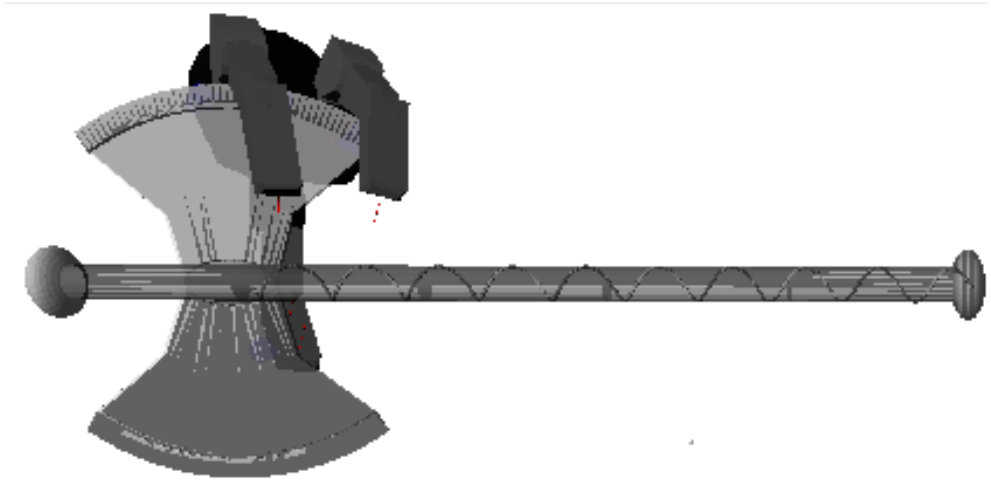
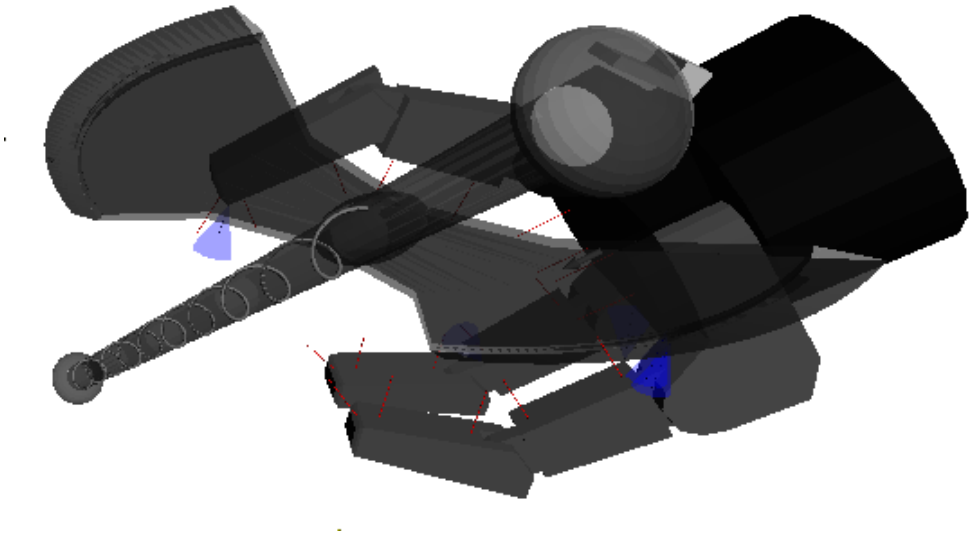


Figure 6.19: Results on grasp search from vision on some test objects.



(a)



(b)

Figure 6.20: Results on grasp search from vision on a cutting tool in the test set. Detail on the joint pinching strategy.

7 CONCLUSIONS

In this thesis we formulate and propose a novel framework for task-oriented grasping based on task dependent metrics to evaluate grasps and use hypotheses in a task-oriented setting. The main advantages of our framework over the current state of the art in task oriented robotic grasping research are:

- it decouples the grasp measurement from the task affordance definition, allowing the use of fixed grasp metrics to evaluate an open set of task affordances
- it allows for the automatic evaluation of grasps in a simulation environment
- being run automatically in a simulation environment it allows the collection of *labelled* grasp data with no human interaction
- by eliminating the need for human intervention in the grasp labeling process it clears the main bottleneck that prevents grasp datasets to scale, allowing for a scalable automatic data collection
- evaluating the grasps objectively it clears the biases that humans have in labeling grasps due to the significant differences between human and robotic hands.

We provide a GraspIt! [18] plugin that implements the data collection of our framework and we showed that we can easily generate millions of labelled grasps on different objects, collecting the two datasets described in Section 6.2 in Table 6.1. Hand-designed affordance functions sufficed for the emergence of smart and unintuitive techniques for grasping for the exemplified tasks of picking, cutting and beating as in Figures 6.9 and 6.10.

From this we extended our framework to be applied with partial information and we experimented with convolutional and PointNet [56] architectures, proposing and testing several variations of the two architectures and a new architecture that we name Local PointNet to learn to infer basic grasp stability and holding hand effort from vision and benchmarked their results, which we consider very promising. Finally, we assembled our learned models into an optimizer that applies our framework to search for the best grasp for picking from vision, with only very partial knowledge of the object to be grasped. The produced grasps on novel objects suggest a great generalization capability of this framework, which is based on the estimation of

physical interactions rather than fixed labels. The models for the task of picking do not only generally produce grasps with force closure, but they also try to locate and wrap the center of mass of objects to the best of their, even partial, knowledge, being able to reproduce also elaborate techniques on novel objects when possible, like joint pinching as in Figure 6.20.

7.1 LIMITATIONS AND FUTURE WORKS

There are many directions in which this work can be improved and extended to account for its limitations. The most obvious directions are by validating on more tasks or by accounting for more metrics in simulation such as different materials in compound objects, with different friction coefficients and softness values. While this may be not relevant for some tasks on some objects, such additional information would also provide more realistic locations for the center of mass, which is crucial to determine the optimal grasp in many other situations such as for the beating with hammers.

Moreover, from the learning side, the current work is limited to learning metrics which are not related with the use location, thus an additional validation step to extend the partial information models to the tasks of cutting and beating is foreseen for the near future.

As a natural follow up of our validation currently limited to benchmarks and simulation, we foresee a validation step on a real robotic arm to prove the generalization capability from simulation to reality. Indeed, we plan to integrate the learned models with a real Barrett hand and test on similar manipulators (with three or even two fingers) to assess the robustness of performance with inaccurate manipulator models.

The current hypothesis of camera-in-hand substantially limits the exploration capability of the algorithm that needs to physically move the arm to evaluate different grasp approach directions. The usage of general point cloud based models allows to easily overcome this limitation as they are a general representation basis for the knowledge on the object geometry. This would not only allow to integrate knowledge from multiple range images (not necessarily from the hand) but also to rotate the object representation to face any hypothesized approach direction to search for more appropriate directions with the current integrated knowledge, without the need of actually moving the cameras if not to gather richer knowledge on the object geometry.

An important limitation of the current work is the definition of the affordance functions $\tilde{F}_T$ which, at this stage, *define* the tasks themselves for the system and must be hardcoded by a human. A direction of improvement is towards a different description of tasks from which functions $\tilde{F}_T$ can be extracted or learned. Reference [50] extracts a representation of a task in terms of relevant physical quantities from the demonstration of a human choosing

a tool and executing the task, such representation can be used to drive the automatic synthesis of the affordance function from a human demonstration. An alternative approach can be the definition of the execution of the task and a performance index of its end effectiveness: this would allow the optimization of the affordance function by reinforcement learning by simulation of the execution of the task itself, again with no further human intervention.

BIBLIOGRAPHY

[1] J. J. Gibson. *The senses considered as perceptual systems*. Boston: Houghton Mifflin, 1966 (cit. on pp. 6, 16).

[2] James J Gibson. "The Ecological Approach to Visual Perception." In: (1979) (cit. on pp. 6, 7, 16).

[3] Jon Barwise. *The situation in logic*. 17. Center for the Study of Language (CSLI), 1989 (cit. on pp. 7, 16).

[4] Ken Shoemake. "Uniform random rotations." In: *Graphics Gems III (IBM Version)*. Elsevier, 1992, pp. 124–132 (cit. on pp. 50, 98).

[5] Alonso H Vera and Herbert A Simon. "Situated action: A symbolic interpretation." In: *Cognitive science* 17.1 (1993), pp. 7–48 (cit. on pp. 7, 16).

[6] Yann LeCun, Yoshua Bengio, et al. "Convolutional networks for images, speech, and time series." In: *The handbook of brain theory and neural networks* 3361.10 (1995), p. 1995 (cit. on pp. 23, 26).

[7] Edward S Reed. *Encountering the world: Toward an ecological psychology*. Oxford University Press, 1996 (cit. on pp. 7, 16).

[8] Marco Santello, Martha Flanders, and John F Soechting. "Postural hand synergies for tool use." In: *Journal of Neuroscience* 18.23 (1998), pp. 10105–10115 (cit. on p. 48).

[9] Andrew T Miller and Peter K Allen. "Examples of 3D grasp quality computations." In: *Proceedings 1999 IEEE International Conference on Robotics and Automation (Cat. No. 99CH36288C)*. Vol. 2. IEEE. 1999, pp. 1240–1246 (cit. on pp. xi, xiii, 1, 35, 39).

[10] WT Townsend. "MCB-industrial robot feature article-Barrett hand grasper." In: *Industrial Robot: An International Journal* 27.3 (2000), pp. 181–188 (cit. on pp. 11, 46).

[11] Rich Caruana, Steve Lawrence, and C Lee Giles. "Overfitting in neural nets: Backpropagation, conjugate gradient, and early stopping." In: *Advances in neural information processing systems*. 2001, pp. 402–408 (cit. on p. 62).

[12] Robert Shaw. "Processes, acts, and experiences: Three stances on the problem of intentionality." In: *Ecological Psychology* 13.4 (2001), pp. 275–314 (cit. on pp. 7, 16).

[13] Mark Steedman. "Plans, affordances, and combinatory grammar." In: *Linguistics and Philosophy* 25.5-6 (2002), pp. 723–753 (cit. on pp. 7, 16).

[14] Anthony Chemero. "An outline of a theory of affordances." In: *Ecological psychology* 15.2 (2003), pp. 181–195 (cit. on pp. 7, 16).

[15] Harry Heft. "Affordances, dynamic experience, and the challenge of reification." In: *Ecological Psychology* 15.2 (2003), pp. 149–180 (cit. on pp. 7, 16).

[16] Claire Michaels. "Affordances: Four Points of Debate." In: *ECOLOGICAL PSYCHOLOGY* 15 (Apr. 2003), pp. 135–148 (cit. on pp. 7, 16).

[17] Thomas A Stoffregen. "Affordances as properties of the animal-environment system." In: *Ecological psychology* 15.2 (2003), pp. 115–134 (cit. on pp. 7, 16).

[18] A. T. Miller and P. K. Allen. "Graspit! A versatile simulator for robotic grasping." In: *IEEE Robotics Automation Magazine* 11.4 (Dec. 2004), pp. 110–122. ISSN: 1070-9932 (cit. on pp. xi, xiv, 2, 6, 39, 45, 103).

[19] Philip Shilane, Patrick Min, Michael Kazhdan, and Thomas Funkhouser. "The princeton shape benchmark." In: *Proceedings Shape Modeling Applications, 2004.* IEEE. 2004, pp. 167–178 (cit. on pp. xi, xiv, 2, 45, 46, 59).

[20] S. Arimoto, R. Ozawa, and M. Yoshida. "Two-Dimensional Stable Blind Grasping under the Gravity Effect." In: *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*. Apr. 2005, pp. 1196–1202 (cit. on p. 9).

[21] S. Arimoto, M. Yoshida, and Ji-Hun Bae. "Stable "blind grasping" of a 3-D object under non-holonomic constraints." In: *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.* May 2006, pp. 2124–2130 (cit. on p. 9).

[22] Michael Kallay. "Computing the moment of inertia of a solid defined by a triangle mesh." In: *Journal of Graphics Tools* 11.2 (2006), pp. 51–57 (cit. on p. 52).

[23] Matei Ciocarlie, Corey Goldfeder, and Peter Allen. "Dexterous grasping via eigengrasps: A low-dimensional approach to a high-complexity problem." In: *Robotics: Science and Systems Manipulation Workshop-Sensing and Adapting to the Real World*. Citeseer. 2007 (cit. on p. 48).

[24] Matei Ciocarlie, Claire Lackner, and Peter Allen. "Soft finger model with adaptive contact geometry for grasping and manipulation tasks." In: *Second Joint EuroHaptics Conference and Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems (WHC'07)*. IEEE. 2007, pp. 219–224 (cit. on p. 53).

[25] Mario Prats, Pedro J Sanz, and Angel P Del Pobil. "Task-oriented grasping using hand preshapes and task frames." In: *Robotics and Automation, 2007 IEEE International Conference on*. IEEE. 2007, pp. 1794–1799 (cit. on pp. 13–16).

[26] Erol Şahin, Maya Çakmak, Mehmet R Doğar, Emre Uğur, and Göktürk Üçoluk. "To afford or not to afford: A new formalization of affordances toward affordance-based robot control." In: *Adaptive Behavior* 15.4 (2007), pp. 447–472 (cit. on p. 16).

[27] CGAL Cgal. *Computational geometry algorithms library*. 2008 (cit. on p. 53).

[28] D. Prattichizzo and J.C. Trinkle. "Grasping." In: *Handbook on Robotics*. Ed. by B. Siciliano and O. Kathib. Springer, 2008, pp. 671–700 (cit. on pp. 6, 52).

[29] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. "Imagenet: A large-scale hierarchical image database." In: *2009 IEEE conference on computer vision and pattern recognition*. Ieee. 2009, pp. 248–255 (cit. on p. 26).

[30] Dae-Jin Kim, Ryan Lovelett, and Aman Behal. "Eye-in-hand stereo visual servoing of an assistive robot arm in unstructured environments." In: *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*. IEEE. 2009, pp. 2326–2331 (cit. on pp. xi, xiii, 1, 12, 35).

[31] Ayşe Naz Erkan, Oliver Kroemer, Renaud Detry, Yasemin Altun, Justus Piater, and Jan Peters. "Learning probabilistic discriminative models of grasp affordances under limited supervision." In: *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*. IEEE. 2010, pp. 1586–1591 (cit. on pp. xi, xiii, 1, 12, 14, 35).

[32] Gaël Guennebaud, Benoit Jacob, et al. "Eigen." In: *URl: http://eigen.tuxfamily. org* (2010) (cit. on p. 50).

[33] Dan Song, Kai Huebner, Ville Kyrki, and Danica Kragic. "Learning task constraints for robot grasping using graphical models." In: *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 2010, pp. 1579–1585 (cit. on pp. 16, 17).

[34] H. Dang, J. Weisz, and P. K. Allen. "Blind grasping: Stable robotic grasping using tactile feedback and hand kinematics." In: *2011 IEEE International Conference on Robotics and Automation*. May 2011, pp. 5917–5922 (cit. on p. 9).

[35] Massimiliano Corsini, Paolo Cignoni, and Roberto Scopigno. "Efficient and flexible sampling with blue noise properties of triangular meshes." In: *IEEE transactions on visualization and computer graphics* 18.6 (2012), pp. 914–924 (cit. on p. 50).

[36] H. Dang and P. K. Allen. "Semantic grasping: Planning robotic grasps functionally suitable for an object manipulation task." In: *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. Oct. 2012, pp. 1311–1317 (cit. on p. 16).

[37] Hao Dang and Peter K Allen. "Tactile experience-based robotic grasping." In: *Workshop on Advances in Tactile Sensing and Touch based Human-Robot Interaction, HRI*. 2012 (cit. on pp. xi, xiii, 1, 9–11, 35).

[38] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "Imagenet classification with deep convolutional neural networks." In: *Advances in neural information processing systems*. 2012, pp. 1097–1105 (cit. on pp. 25, 26).

[39] Lavanya Sharan, Ce Liu, Ruth Rosenholtz, and Edward H Adelson. "Recognizing materials using perceptually inspired features." In: *International journal of computer vision* 103.3 (2013), pp. 348–371 (cit. on p. 13).

[40] Hao Dang and Peter K Allen. "Semantic grasping: planning task-specific stable robotic grasps." In: *Autonomous Robots* 37.3 (2014), pp. 301–316 (cit. on p. 16).

[41] Diederik P Kingma and Jimmy Ba. "Adam: A method for stochastic optimization." In: *arXiv preprint arXiv:1412.6980* (2014) (cit. on p. 62).

[42] Tuomas J Lukka, Timo Tossavainen, Janne V Kujala, and Tapani Raiko. "ZenRobotics Recycler-Robotic sorting using machine learning." In: *Proceedings of the International Conference on Sensor-Based Sorting (SBS)*. 2014 (cit. on pp. xiv, 1).

[43] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. "Dropout: a simple way to prevent neural networks from overfitting." In: *The Journal of Machine Learning Research* 15.1 (2014), pp. 1929–1958 (cit. on p. 62).

[44] Sean Bell, Paul Upchurch, Noah Snavely, and Kavita Bala. "Material recognition in the wild with the materials in context database." In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 3479–3487 (cit. on p. 13).

[45] Sergey Ioffe and Christian Szegedy. "Batch normalization: Accelerating deep network training by reducing internal covariate shift." In: *arXiv preprint arXiv:1502.03167* (2015) (cit. on p. 32).

[46] Austin Myers, Ching L Teo, Cornelia Fermüller, and Yiannis Aloimonos. "Affordance detection of tool parts from geometric features." In: *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2015, pp. 1374–1381 (cit. on p. 16).

[47] Tu-Hoa Pham, Abderrahmane Kheddar, Ammar Qammaz, and Antonis A Argyros. "Towards force sensing from vision: Observing hand-object interactions to infer manipulation forces." In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2015, pp. 2810–2819 (cit. on p. 13).

[48] Máximo A Roa and Raúl Suárez. "Grasp quality measures: review and performance." In: *Autonomous robots* 38.1 (2015), pp. 65–88 (cit. on pp. xi, xiii, 1, 35).

[49] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. "Going Deeper With Convolutions." In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2015 (cit. on pp. 26, 27, 66).

[50] Yixin Zhu, Yibiao Zhao, and Song Chun Zhu. "Understanding tools: Task-oriented object modeling, learning and recognition." In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2015, pp. 2855–2864 (cit. on pp. 14, 15, 104).

[51] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Deep residual learning for image recognition." In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778 (cit. on pp. 28, 30, 66).

[52] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. "Rethinking the Inception Architecture for Computer Vision." In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2016 (cit. on pp. 27, 29).

[53] François Chollet. "Xception: Deep learning with depthwise separable convolutions." In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 1251–1258 (cit. on pp. 31, 32, 66, 70, 90).

[54] Renaud Detry, Jeremie Papon, and Larry Matthies. "Task-oriented grasping with semantic and geometric scene understanding." In: *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2017, pp. 3266–3273 (cit. on pp. 16, 17).

[55] GL Manoj Kumar and Tim Head. "Scikit-optimize." In: *Tim Head and contributors* (2017) (cit. on p. 93).

[56] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas 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 pp. xiv, 2, 31, 34, 39, 66, 67, 103).

[57] Xinhang Song, Luis Herranz, and Shuqiang Jiang. "Depth CNNs for RGB-D scene recognition: learning from scratch better than transferring from RGB-CNNs." In: *Thirty-First AAAI Conference on Artificial Intelligence*. 2017 (cit. on p. 67).

[58] Trevor Standley, Ozan Sener, Dawn Chen, and Silvio Savarese. "image2mass: Estimating the Mass of an Object from Its Image." In: *Conference on Robot Learning*. 2017, pp. 324–333 (cit. on p. 13).

[59] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alexander A Alemi. "Inception-v4, inception-resnet and the impact of residual connections on learning." In: *Thirty-First AAAI Conference on Artificial Intelligence*. 2017 (cit. on pp. 30, 31, 66).

[60] Philipp Zech, Simon Haller, Safoura Rezapour Lakani, Barry Ridge, Emre Ugur, and Justus Piater. "Computational models of affordance in robotics: a taxonomy and systematic classification." In: *Adaptive Behavior* 25.5 (2017), pp. 235–271. eprint: https://doi.org/10.1177/1059712317726357 (cit. on p. 6).

[61] Sergey Levine, Peter Pastor, Alex Krizhevsky, Julian Ibarz, and Deirdre Quillen. "Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection." In: *The International Journal of Robotics Research* 37.4-5 (2018), pp. 421–436 (cit. on pp. xi, xiii, 1, 11, 12, 35).

[62] Andy Zeng, Shuran Song, Kuan-Ting Yu, Elliott Donlon, Francois R Hogan, Maria Bauza, Daolin Ma, Orion Taylor, Melody Liu, Eudald Romo, et al. "Robotic pick-and-place of novel objects in clutter with multi-affordance grasping and cross-domain image matching." In: *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2018, pp. 1–8 (cit. on pp. xi, xiii, 1, 12, 13, 35).

[63] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. "Open3D: A modern library for 3D data processing." In: *arXiv preprint arXiv:1801.09847* (2018) (cit. on p. 62).

[64] Safoura Rezapour Lakani, Antonio J. Rodríguez-Sánchez, and Justus Piater. "Towards affordance detection for robot manipulation using affordance for parts and parts for affordance." In: *Autonomous Robots* 43.5 (June 2019), pp. 1155–1172. ISSN: 1573-7527 (cit. on p. 16).

Part I

APPENDIX A

Towards Affordance Prediction with Vision via Task Oriented Grasp Quality Metrics

Luca Cavalli
Politecnico di Milano
Milan, Italy
luca3.cavalli@mail.polimi.it

Gianpaolo Di Pietro
Politecnico di Milano
Milan, Italy
gianpaolo.dipietro@mail.polimi.it

Matteo Matteucci
Politecnico di Milano
Milan, Italy
matteo.matteucci@polimi.it

***Abstract*—While many quality metrics exist to evaluate the quality of a grasp by itself, no clear quantification of the quality of a grasp relatively to the task the grasp is used for has been defined yet. In this paper we propose a framework to extend the concept of grasp quality metric to task-oriented grasping by defining affordance functions via basic grasp metrics for an open set of task affordances. We evaluate both the effectivity of the proposed task oriented metrics and their practical applicability by learning to infer them from vision. Indeed, we assess the validity of our novel framework both in the context of perfect information, i.e., known object model, and in the partial information context, i.e., inferring task oriented metrics from vision, underlining advantages and limitations of both situations. In the former, physical metrics of grasp hypotheses on an object are defined and computed in known object model simulation, in the latter deep models are trained to infer such properties from partial information in the form of synthesized range images.**

***Index Terms*—task-oriented grasping, robotic grasping, affordance, vision**

I. Introduction

The research community has spent much effort in tackling the problem of grasping novel objects in different settings [1] [2] [3] [4] [5] with the objective of holding objects robustly with robotic manipulators; however, real manipulation tasks go far beyond holding the objects and the quality of a grasp depends on the task it is meant to support. While many quality metrics exist to evaluate the quality of a grasp by itself [6] [7], no clear quantification of the quality of a grasp relatively to a task has been defined. In this paper we propose a framework to extend the concept of quality metric to task-oriented grasping by defining general physical measures for an open set of task affordances. We evaluate both the results provided by such metrics and their applicability in practice by learning to infer them from vision.

More formally, given a grasp G on an object O and a point U on the surface of O (which is the point where we plan to use the object, when the task requires one), we define the affordance function $F_T : (O, G, U) \mapsto \mathbb{R}$ to define the affordance of any possible grasp G and use hypothesis U with respect to task T. The final objective is to optimize for the best grasp, object and use location (O, G, U) that maximizes a given affordance function, as shown in Figure 1. We are interested in finding a set of metrics, as functions of the triplet (O, G, U), to encode significant static and geometric properties of the (O, G, U) system itself. Examples of such metrics are the local geometry of O around U, or the minimum sum of contact forces needed to hold the object O with grasp G under a given gravity vector. A complete description of the metrics used in this work, not to be considered as an exhaustive list, is reported in Section IV-A.

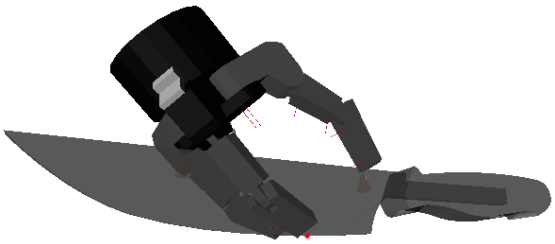


Fig. 1. Best grasp, according to the proposed metrics, relatively to the affordance function $\tilde{F}_{cut}$ defined in Section IV-B for the cutting task using a common kitchen knife

The proposed approach allows the inference of the affordance of objects without having it bound to their semantic category: semantic information on objects defines their standard use meant for humans, which is not necessarily the only nor even optimal use for robots. Semantics greatly simplifies the task of affordance perception, but it gives no guarantee of optimality, particularly with robot actuators which differ substantially from human hands and arms. Take, for instance, the classical human grasp of a hammer with the wrist direction parallel to the beating direction: actuating such grasp with a Barrett hand [8] which has only a rotational degree of freedom on the wrist would have the same efficacy in beating as a human with a locked wrist, even without taking into consideration the decreased number of fingers and much reduced tangential and torsional friction in contacts.

To validate our framework, we have collected a dataset of grasps and computed elementary graps metrics by using the GraspIt! simulator [9] on the Princeton Shape Benchmark object models [10]. Then we have trained deep models to infer those elementary metrics from range images taken in a simulated camera-in-hand setting to assess the applicability of our framework to more realistic partial-information settings.

In our contribution, we define a framework for quality assessment of task-oriented grasps, we qualitatively validate such framework, we provide a GraspIt! plugin to produce

labelled data with minimal to no human intervention, thus in an extremely scalable way, and we generated a dataset of 400M evaluated grasps on 22 objects of the Princeton Shape Benchmark which we plan to make public in the near future. Moreover, we propose and benchmark models to tackle the problem of learning to infer such metrics from vision. Preliminary results shows direct optimization of affordance functions in simulation produces new and creative grasps which fit the specific actuator in use for the selected task, while direct inference of such metrics from vision is yet an open challenge and there are great margins for further improvement.

II. Related works

Many researchers have worked towards the understanding and formalization of the concept of affordances [11] [12] [13]; they have been inspiring for roboticists to work within the affordances framework to define the autonomous interaction of a robot with an unknown environment. In our work we investigate the broad category of robot affordances focusing on the specific application of task-oriented grasping. Within this context, one of the first approaches towards task-oriented grasping, reported in [14], proposed to encode the task in physical terms (e.g., applying a momentum on a handle to open a door) and then to solve the problem of grasp planning by hardcoding hand postures and their association with tasks; the method has shown good performance in the expected domain, but poor generalization capabilities. Later works formalized the problem via graphical models, distinguishing task, object features, action features and constraint features. In particular, authors of [15] proposed the use of such formalization and they have been able to effectively learn to infer the likelihood of grasp approach directions with respect to a human-labelled ground truth. The main limitation of this work, in our opinion, is the human intervention, which makes the real definition of the tasks implicit and prevents the scalability of the dataset that can be generated for learning without tedious human teaching. The direct intervention of human judgment on semantics to evaluate the quality of grasp hypotheses with respect to a given task is nevertheless a common approach to many research works, like [16] and [17] in which authors prove the effectiveness of a human-labelled semantic approach with real robot manipulations. More recently, [18] has proposed to label mesh vertices in simulated objects as being graspable or not according to some task, so that many scene examples can be produced and automatically labeled via simulation. This allowed the system to automatically segment graspable and not graspable regions of objects in cluttered scenes, but still the expressivity of this method is restricted to specifying graspable or not graspable surfaces. Reference [19] proposed a bottom-up approach for affordance perception by object parts which detects the local geometry of patches of the object and provides pixel-level affordance segmentation for pre-defined affordance classes. Their method is again based on a dataset [20] of 10000 pixel-labelled RGB-D images which have been hand labelled.

A common limitation of these approaches is the vague definition of task affordances which passes through the human labeling of the ground truth. This entails a great limitation in the size of the data that can be produced for learning and poses questions about the optimality and validity of the labels with respect to the actual task performance with a different actuator than the human hand.

III. Proposed approach

Let $\mathcal{O}$ be the set of possible object surfaces with friction and softness properties defined at each point, let $\mathcal{G}(O)$ defined on an object $O \in \mathcal{O}$ be the set of possible grasps determined by the hand embodiment, degrees of freedom, contact locations on the object and contact nature (e.g., frictionless, hard contact or soft contact), and $\mathcal{U}(O)$ be the set of points on the surface of the same object that can be considered as points of use.

Let $O \in \mathcal{O}$, $G \in \mathcal{G}(O)$, $U \in \mathcal{U}(O)$, then we define the *affordance function* of task T as $F_T(O, G, U) \mapsto \mathbb{R}$ such that $F_T(O_1, G_1, U_1) > F_T(O_2, G_2, U_2)$ if and only if the grasp and use hypothesis (O_1, G_1, U_1) is more suitable than the hypothesis (O_2, G_2, U_2) for task T, thus defining an affordance ordering of an object grasp for task T.

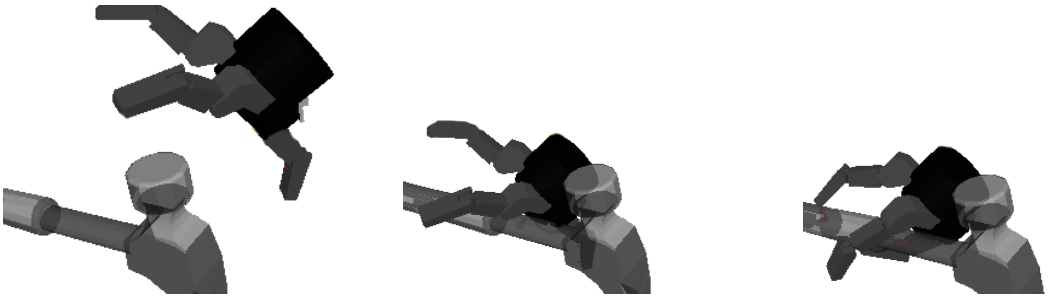
As we want a compact representation of the affordance function, we approximate F_T as a $\tilde{F}_T : \mathbb{R}^n \mapsto \mathbb{R}$ by mapping the triplet (O, G, U) into a metric vector $\phi \in \mathbb{R}^n$ through a function $\Phi(O, G, U) \mapsto \mathbb{R}^n$. This metric vector is a collection of metrics encoding the geometrical and static physical properties of the triplet (O, G, U) which are relevant to approximate F_T. In Section IV we provide some examples of basic metrics ϕ and examples on how they could be used to hardcode $\tilde{F}_T$ for some reference tasks.

A. Achieving Object Semantics Independence

The complete object geometry is generally not available in real world applications, in particular when our long term goal is to infer object affordance from vision with no hardwired semantics. To achieve such goal, we need to frame the problem in the context of uncertain and incomplete information about the object by decoupling the grasp and use location description from the exact object geometry and possibly its semantics.

Recall here that the complete description of a grasp requires the geometry and nature of contact points on the grasped object, and the grasp itself needs to be actuated by a grasping policy. If we assume the grasping policy to be deterministic, then we can define it as a function $GP(p_0, O) \mapsto \mathcal{G}(O)$ that maps an initial state $p_0 \in \mathcal{P}_0$ and an object O into the final grasp $G \in \mathcal{G}(O)$. To decouple from the specific grasp, and its parameters, we fix a grasping policy that allows a sufficient exploration of the grasps space $\mathcal{G}$ via the space $\mathcal{P}_0$ of possible initial states, which we call *pregrasps*.

In particular, we select a simple, but effective, grasping policy defined as follows: from an initial position of the hand with open fingers, we advance towards a fixed direction until the first contact is made, then the fingers are closed until all of them either make contact or are completely closed. Jointly with the use of eigengrasps [21] to describe the degrees of freedom



(a) (b) (c)

Fig. 2. The three phases of our grasp policy: (a) the pregrasp parameters determine the initial position and posture of the hand (b) the hand approaches in a straight line until a contact is made (c) fingers close until they make contact or they are completely closed

of the hand, this policy allows for a further reduction in the dimensionality of the problem for the translational degree of freedom saved in describing the position of the hand in the space, as approaching the object in a straight line makes the approach direction invariant with respect to the policy. If the eigengrasp parameters are divided into h parameters used to close the grasp and k parameters used to set the initial hand posture, then we can consider $p_0 \in \mathbb{R}^{5+k}$, where the 5 extra parameters describe the initial position and rotation of the hand. The h eigengrasp parameters used to close the hand are fixed to an open position on the pregrasp, thus they do not contribute to the dimensionality of the pregrasp itself.

On the other hand, use locations are points on the bidimensional surface of each specific object; we decouple them from the specific object by defining use directions and a direction mapping function $DM(d, O) \mapsto \mathcal{U}(O)$ in complete analogy to the grasp decoupling solution. Our direction mapping assumes $d \in \mathbb{R}^2$ to be the spherical coordinates of a directed ray centered in the center of mass of object O and output in the farthest point $U \in \mathcal{U}(O)$ which is the intersection of such ray with the outer object surface.

B. Metrics Inference from Vision

As our goal is to make robots able to use unknown objects in a task consistent way, we need the robot to be able to perceive their affordances via sensors. In particular, we focus on vision being it an extremely common and effective tool to take information from the environment in real applications. We define our inference setting by focusing on the specific case of single range images taken from camera-in-hand perspective: such case provides only local geometry information about the object around the expected location of the planned grasp. We want to learn a model that predicts the elementary metrics $\phi \in \mathbb{R}^n$ of a triplet (O, G, U) with only partial information about the observed object O. Indeed, predicting the vector ϕ would allow to estimate the affordance function F_T for any task T for which we can define an $\tilde{F}_T$.

Let $D : \mathcal{P}_0 \times \mathcal{O} \mapsto \mathbb{R}^{h \times w}$ be the function that maps a pregrasp p_0 on an object O to the depthmap of size $h \times w$ from the camera-in-hand perspective of p_0. Then we want to learn a model $\mathcal{M}^\Phi$ that approximates the mapping from $(p_0, D(p_0, O), d)$ to $\Phi(O, GP(p_0, O), DM(d, O))$ where O is an object, p_0 is a pregrasp, d is a use direction, and Φ is the metric extraction function, under the grasping policy GP and the direction mapping function DM.

We assume to be able to learn model $\mathcal{M}^\Phi$ from a dataset of tuples $(O, p_0, d, \Phi(O, GP(p_0, O), DM(d, O))$ obtained via uniform sampling of p_0 and d values on a set of available objects models and computing the true values of $\Phi(O, GP(p_0, O), DM(d, O))$ via simulation. Details on our data collection setup are explained in Section V-A.

To structure the learning task, we define the model $\mathcal{M}^\Phi$ as the composition of two models: an input value $(p_0, D(p_0, O), d)$ is first classified by a binary classifier $\mathcal{M}^\Phi_C$ to infer whether it represents a "good" grasp worth further evaluation or not. We define "good" grasps those respecting a minimum quality independently from the task, thus employing state of the art grasp quality metrics to generate the ground truth. The samples classified as positive then pass through a regression model $\mathcal{M}^\Phi_R$ that infers the metrics ϕ with the implicit assumption that the grasp is indeed a quality grasp.

For both models $\mathcal{M}^\Phi_C$ and $\mathcal{M}^\Phi_R$ we propose and evaluate the two architectures of the convolutional neural network (CNN) and the PointNet architecture [22]. Both architectures encode the available geometry information (in the form of range image for the CNN or as the equivalent projected point cloud for the PointNet) in a feature vector, we then apply late-fusion of the other input parameters p_0 and d on this feature vector and output the classification label or the inferred regression value with a classical fully connected network.

IV. Task Oriented Grasp Metrics

In this work we concentrate on the sample tasks of beating, cutting and picking, which are defined by their $\tilde{F}_{beat}$, $\tilde{F}_{cut}$ and $\tilde{F}_{pick}$ in Section IV-B. Such tasks have been selected with the idea of the kitchen assistant robot in mind, considering some very different tasks that may happen to be requested in this sample application. We first define a set of grasp metrics, then define from these the corresponding affordances.

A. Basic Grasp Metrics

We consider the following set of elementary metrics of (O, G, U) which should not be considered as exhaustive:

a) Grasp robustness ($\epsilon \in \mathbb{R}$): is a real number describing the robustness of the grasp. We use the Epsilon metric described in [7] as a builtin in the GraspIt! simulator [9]. Force closure grasps have $\epsilon > 0$ and higher robustness implies a greater minimum perturbance is needed to break the grasp.

b) Rotational inertia ($I \in \mathbb{R}$): quantifies the rotational inertia around the axis of rotation of the wrist of the hand assuming a unitary density of the object and assuming the hand to be integral with the whole object. It does not take into account the mass of the hand itself.

c) Hand effort on impact ($E_i \in \mathbb{R}$): describes the effort of the hand to balance the impact forces after a rotation around the wrist. It assumes a fixed average inertial torque in a small Δt during the impact which is directly proportional to I and a free contact force on the use location towards the normal direction. This metric takes the value of the minimum sum of the contact forces of the hand constrained to the contact friction cones to balance the inertial torque, ∞ if the minimization problem is unfeasible.

d) Hand effort on hold ($E_h \in \mathbb{R}^6$): is a vector of six independent values which quantify the hand effort to balance a different gravity vector. The hand effort is the minimum sum of all contact forces constrained to the contact friction cones that balance a given unitary force of gravity, ∞ if such problem is unfeasible. The six gravity vectors chosen are aligned with the three coordinate axes (once in the same direction, once opposite) of the object mesh as all meshes that we used in the Princeton Shape Benchmark have been designed by humans that gave a semantic meaning to the coordinate axes directions.

e) Momentum discharge efficiency ($\delta \in \mathbb{R}$): quantifies the efficiency of discharging the rotational inertia of the wrist on the object use location. It quantifies the alignment between the inertial torque and the torque generated by a force aligned with the use location normal vector towards the inside of the object surface. It is computed as the dot product of the two normalized vectors, clipped to zero in case of negative values.

f) Force transmitted to use ($U_\tau \in \mathbb{R}$): quantifies the force that can be transmitted to the use location using constrained contact forces. It assumes all contact forces are constrained by their friction cones and have unitary maximum normal forces. It takes the value of the maximum force on the use location towards the use location normal guaranteeing static conditions.

g) Use local geometry ($U_g \in \mathbb{R}$): describes how much the use location has the shape of an edge. It is obtained by fitting a quadratic function on the vertices of the triangles near the use location (including all the triangles that share at least one vertex with the triangle where the use location lies) and extracting the eigenvalues of the hessian matrix of such quadratic function. The two eigenvalues λ_1 and λ_2 are the two principal component curvatures, so we quantify an edge with the expression $(\lambda_1 - \lambda_2)^2$ to identify locations with a great difference in local curvatures.

B. Affordance Functions from Basic Metrics

On top of these metrics we define the affordance functions for $T \in \{beat, cut, pick\}$. In this preliminary study affordance functions have been designed by hand, to validate the feasibility of the framework, in future works we aim to learn them by optimizing task execution efficacy.

a) Beating: The classical beating action of a hammer with a human hand requires dexterous movements of the wrist which would need moving the whole robotic arm to be reproduced on a Barrett hand. For this reason we assume that the beating action will be executed by the robotic actuator by simply rotating the hand clockwise around the wrist. We require that the hold is stable over a minimum threshold and that the rotational energy gets discharged almost entirely on the point of use. We want to maximize the ratio of the energy that we can incorporate into the rotation (assuming a maximum rotational speed) over the actual hand effort of keeping the object stable on the impact.

Input: $\epsilon, \delta, I, E_i, E_h$
Output: $\tilde{F}_{beat}$
1: **if** $\epsilon < \tau_\epsilon$ OR $\delta < \tau_\delta$ OR $\sum_{i=1}^{6} E_h[i] == \infty$ **then**
2: **return** $-\infty$
3: **else**
4: **return** $\frac{I}{E_i}$
5: **end if**

b) Cutting: The action of cutting is extremely complex by itself and varies greatly with different materials and their surface and micro-structural properties. A complete physical study of this particular task is not our objective; we simplify it considering as approximation that greater force provides cuts if executed on a thin enough edge.

Input: ϵ, U_τ, U_g
Output: $\tilde{F}_{cut}$
1: **if** $\epsilon < \tau_\epsilon$ OR $U_g < \tau_{U_g}$ **then**
2: **return** $-\infty$
3: **else**
4: **return** U_τ
5: **end if**

c) Picking: Picking an object (as the first part of the pick-and-place task) only strictly requires a stable grasp for a successful pick. However, different stable grasps may imply very different effort from the hand actuator to balance the force of gravity on the object. For this reason we require a stable grasp and minimize the sum of the contact forces required to balance the force of gravity in the six directions evaluated by the E_h metric. Notice that an unstable grasp will need to have at least one evaluated direction of gravity that the grasp cannot hold, thus we do not check the ϵ metric.

Input: E_h
Output: $\tilde{F}_{pick}$
1: **return** $-\sum_{i=1}^{6} E_h[i]$

(a) (b) (c)

Fig. 3. Pregrasp degree of freedom (a) set to 0, (b) set to 0.25, (c) set to 1

V. Framework Validation

To assess the feasibility of the proposed approach we performed a set of experiments focused on the tasks of beating, cutting and picking; this choice has driven the selection of basics metrics to encode in function Φ and the definition of functions $\tilde{F}_{beat}$, $\tilde{F}_{cut}$ and $\tilde{F}_{pick}$ in the previous section. In the validation we aim at:

1) **Validating the framework** by showing that $\text{argmax}_{G,U}\, \tilde{F}_T(\Phi(O,G,U))$ for some selected tasks T provides grasps and use locations that are semantically meaningful with respect to the semantics of task $T \in \{beat, pick, cut\}$
2) **Assessing the feasibility of learning** a model $\mathcal{M}^{\Phi}$ that can infer basic grasp metrics from partial information about a target object.

We provide an implementation of the metric extraction function Φ for the selected metrics (which we describe in Section IV-A) as a plugin for the GraspIt! [9] simulator, some of which are formulated as linear programming problems which we solve through the CGAL library [23]. Input object models are selected from the Princeton Shape Benchmark [10] dataset and assumed to be constituted of homogeneous plastic, and grasps are produced using the model of a Barrett hand [8]. All simulated grasps are evaluated and results are logged into a dataset, preserving both stable and unstable grasps. We structure the model $\mathcal{M}^{\Phi}$ as a classifier that filters stable grasps only and a regressor that estimates the metric vector ϕ of stable grasps from the available partial information.

A. Data collection

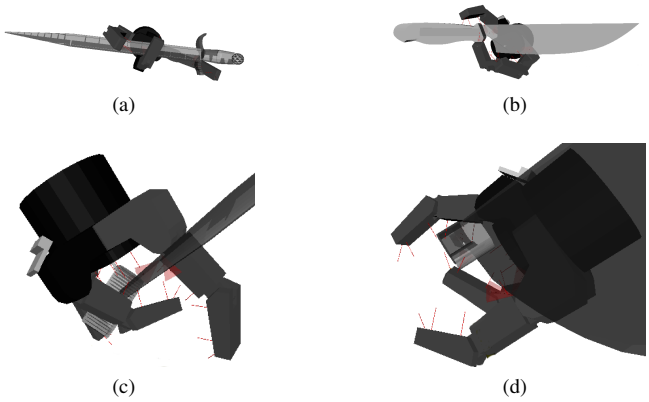
We collected a dataset of grasp and use hypotheses to compute metrics for learning purposes. We sample pregrasps and use directions with uniform distribution in their domain and then simulate the grasp and determine the exact use location on the mesh of an object from the Princeton Shape Benchmark [10]. The grasp is simulated with GraspIt! [9] on a Barrett hand [8] with the policy in Figure 2: from an initial hand position and orientation, the manipulator advances in a straight line until a first contact is made with the object, then the three fingers of the Barrett hand are closed independently until a contact is made or the finger is completely closed.

B. Optimization of Task Grasp Metrics via Simulation

We consider only one pregrasp degree of freedom for the Barrett hand to encode the angle between the two joint fingers as shown in Figure 3, thus the domain in which we uniformly randomize the pregrasps is $[-1,1]^5 \times [0,1]$: two values in $[-1,1]$ encode the hand approach direction in normalized spherical coordinates, one value in $[-1,1]$ encodes the hand rotation around its approach axis, two values in $[-1,1]$ are the approach offset on the xy-plane relatively to the bounding box of the considered object and one value in $[0,1]$ is the pregrasp degree of freedom of the Barrett hand; use directions are encoded in normalized spherical coordinates in $[-1,1]^2$. Data collection can be run in parallel on multiple cores and machines, producing millions of data samples each day. Running on 20 cores of an Intel Xeon E5-2630 v4 for a week we could produce a dataset of over 400 million samples of random grasps with metrics, out of which 20 million samples are viable grasps which respect the condition in Section V-C.

TABLE I
Affordance function thresholds

Mode name	τ_ϵ	τ_{U_q}	τ_δ
Default	0.3	10	0.95
No robustness required	$-\infty$	10	0.95
Extra robustness required	0.5	10	0.95



(a) (b) (c) (d)

Fig. 4. Optimized grasps from the picking showing the joint pinch strategy. The edge of the blade is pinched between two joints to improve the stability of the grasp; (c) and (d) show the detail of the joint pinch on the blade.

To assess the validity of our framework we extract $\text{argmax}_{G,U}\, \tilde{F}_T(\Phi(O,G,U))$ for our three selected tasks by brute force search on our dataset samples. The parametric thresholds used are the default values reported in Table I. Sample results of this procedure are shown in Figures 4, 5, 6, and 7. Some of the produced grasps do not appear intuitive, such as the grasps produced for moving blades in Figure 4, because they exploit features of their physical actuator which are very different from a human hand. In this particular case we observe that the hand achieves a stable grasp by pinching the edge of the blade between the joint of some finger. This pinch provides multiple contacts with very different normals that provide much greater stability of the grasp on a low friction material. The emergence of such solution is very unlikely to happen from human evaluation of grasps, as the human intuition is heavily biased by the human hand with much more fingers, much higher tangential and torsional

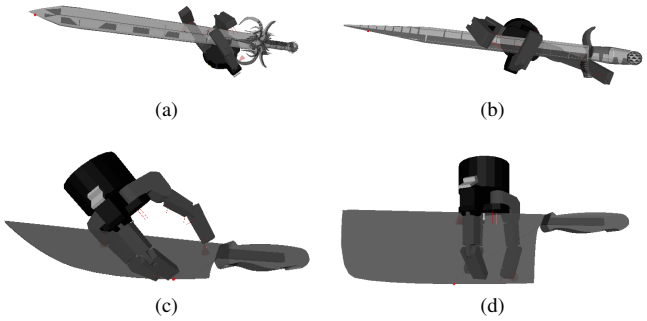


Fig. 5. Optimized grasps from the cutting task showing two different cutting strategies: thin blade tools like (a) and (b) exercise pressure by rotation, wide blade tools like (c) and (d) instead prefer a direct pressure strategy.

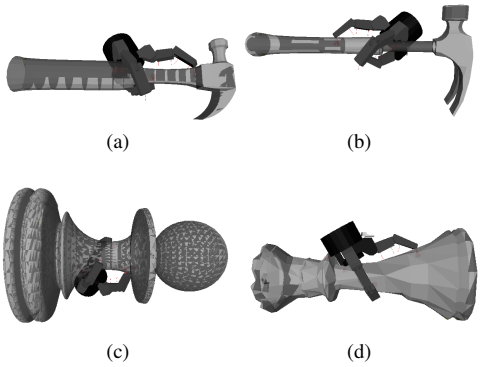


Fig. 6. Optimized grasps from the picking task on sample objects.

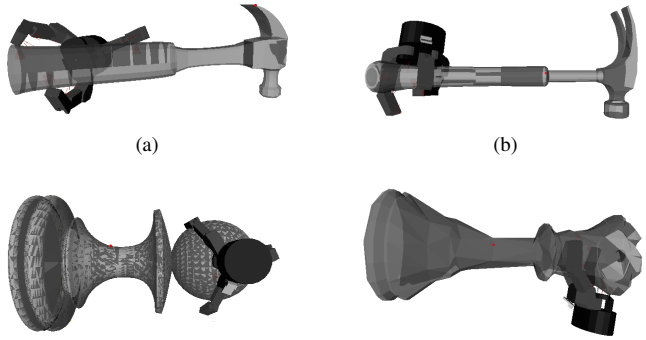


Fig. 7. Optimized grasps from the beating task on sample objects. Notice that the center of mass of hammers in (a) and (b) is on the handle, as the material is assumed homogeneous.

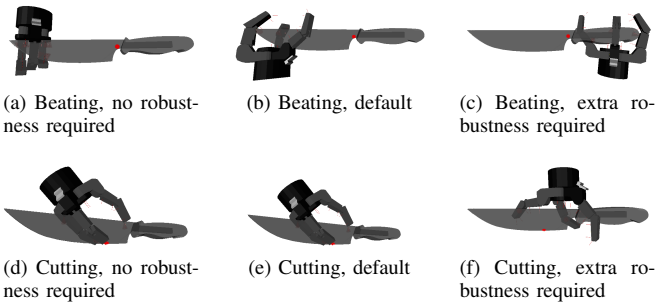


Fig. 8. Affordance function parameter variability. The grasp strategy varies slightly while changing the requirement of the robustness of grasp.

friction and more susceptible to damage than the metal Barrett hand assumed in our experiments.

Adaptive behaviours are evident in Figure 5: with thin blades the hand exercises pressure on the edge by rotation, using the handle as a fulcrum, while with larger blades where such technique is not feasible a direct pressure is preferred. Picking grasps (Figure 6 and 4) generally wrap around the center of mass as a direct result of the minimization of the total contact forces for holding against gravity. Beating grasps (Figure 7) display greater variance and generally achieve a stable grasp on the object far from the center of mass to increase the rotational inertia of the object and choose a use location very well aligned with the rotation direction to effectively discharge the rotational energy on the target (the values of δ for the optimal grasps are far nearer to 1 than the required values for the selected threshold τ_δ). The use location is selected slightly off the center of mass from the opposite side of the hand to balance the beating impulse and produce a torque that contrasts the rotational inertia of the beating movement.

As a further validation test we changed the ϵ threshold requirement for the tasks of beating and cutting according to Table I. The results of such experiment conducted on the model of a typical kitchen knife are shown in Figure 8. The produced grasps for beating with a knife display a similar strategy with respect to the ones for more classical objects as seen in Figure 7: the grasp is as far as possible from the center of mass, switching to the handle only when greater robustness is required. The cutting task on the knife shows less variance, producing robust grasps even when not explicitly required by the fitness function.

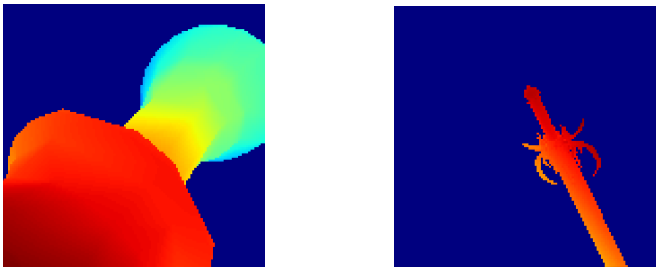
C. Learning Grasp Metrics from Vision

For the learning phase we defined exactly what a viable grasp should be to generate the ground truth for the classifier network and define the actual dataset for the regressor. We define a viable grasp sample if:

$$\epsilon > \tau_\epsilon \quad \wedge \quad \sum_{i=1}^{6} E_h[i] < \tau_{E_h} \quad \wedge \quad E_i < \tau_{E_i}$$

where empirically we set $\tau_\epsilon = 0.15$, $\tau_{E_h} = 250$, $\tau_{E_i} = 100$.

Range images are generated using OpenGL with a perspective projection using common parameters. We take the Kinekt 2 depth camera as a reference with a 70x60 field of view angle: we generate subsampled images of resolution 128x128 using a 60 field of view taken from an object-length distance from the center of the object, as shown in Figure 9. We formally consider point clouds equivalent to range images as they are generated to hold the same exact



(a) (b)

Fig. 9. Synthesized range images with camera-in-hand perspective.

TABLE II
RESULTS ON REGRESSORS

Model	MSE	Comparison accuracy	Expected GMS
PointNet Full	0.050	0.63	0.63
PointNet Slim	0.049	0.60	0.73
CNN	0.049	0.66	0.82

information as the input image. We use Open3D [24] to project the synthesized range image to a point cloud which is cleaned from the background. The resulting point cloud is either randomly subsampled of filled with extra points in the origin to match a standard number of points to build batches for efficient learning. We selected the training, validation and test set from the Princeton Shape Benchmark to propose examples of very different geometries in the training set (1 hammer, 1 screwdriver, 2 bottles, 1 glass, 1 sword, 1 dagger, 1 meat cleaver, 2 ice creams) and to propose similar interesting semantic categories in the validation (1 hammer, 1 bottle, 1 sword, 1 dagger, 1 knife) and test (1 screwdriver, 1 bottle, 1 axe) sets. Validation and test comprise different objects from train but with overlapping semantic categories.

a) The Classifier: filters viable grasps from input data based on the pregrasp and on the input range image with camera-in-hand perspective. As the overall data distribution from our random policy is highly biased towards the negative class (20 times more likely than the positive class), we sample the training data to balance positive and negative samples.

b) The Regressor: infers the metric vector ϕ for later computation of the affordance function relative to the input sample. In this work we trained regressor networks to infer the value of $\sum_{i=1}^{6} E_h[i]$ for testing with the picking affordance function on the picking task. Output values are linearly normalized in the interval $[0, 1]$ from the original domain $[0, \tau_{E_h}]$ granted by the assumption that input values come from the positive class of viable grasps.

The precision-recall curve of the test set for the two trained classifiers are reported in Figure 10. As this classifier model is intended as a filter of good grasp hypotheses, our main metric of interest is the precision of the positive class as this represents the probability that a grasp that passes the filter does really satisfy the expected stability conditions. The recall of the positive class is relevant as well, as it describes the efficiency of the system in missing less good grasp.

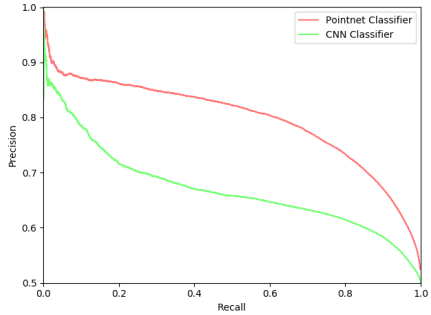


Fig. 10. Recall-precision curve for CNN and PointNet classifiers. Curves are drawn on the performance on the test set.

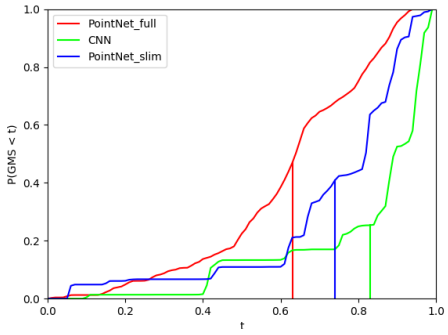


Fig. 11. Global minimum score cumulative distributions for the three regressors. The PointNet full regressor is the standard PointNet, the PointNet slim is the standard PointNet with transformation layers fixed to the identity. The vertical lines of each distribution is the expected value for the GMS.

The results on the test set of the trained regressors are in Table II. The mean squared error, the classical metric used to assess basic regression, gives an overall score of how near the regression goes to real values, but as our goal is optimization, not estimation, we elaborated on two different metrics. As optimization is mainly built by comparisons, we elaborate the comparison accuracy by sampling random couples of samples and measuring the standard accuracy in telling which input sample corresponds to a greater value of the metric. We define the Global Min Score (GMS): let I_{min} be the input sample that minimizes the predicted output $\mathcal{M}_R^{\Phi}(I_{min})$ over a set S of samples, then the Global Min Score of the model for the set S is the rate of samples that actually have a greater ground truth value than the ground truth value of I_{min}. As this score is very sensible to different choices of S, we sample random subsets of 10% of the available samples in S to plot the probability distribution of the value of GMS. Figure 11 shows the cumulative GMS distributions and their respective expected value. This is the most specialized measure of performance of our regressor models as it directly quantifies the relative optimality of the selection of the model relatively to the available choices.

VI. Discussion on Limitations and Future Work

In this work we proposed task dependent metrics to evaluate grasps and use hypotheses in a task-oriented setting. Our

framework allows for the automatic evaluation of grasps in a simulation environment and to collect labelled grasp data with minimal human interaction; eliminating the need for human intervention in the grasp labeling process allows both for a widely more scalable data collection and clears the biases that humans have in labeling grasps due to the significant differences between human and robotic hands. We showed that we can easily generate millions of labelled grasps on different objects and that roughly hand-designed affordance functions suffice for the emergence of smart and unintuitive techniques for grasping for the exemplified tasks as in Figures 4 and 5. From this we experimented with convolutional and PointNet [22] architectures to learn to infer basic grasp stability and holding hand effort from vision and benchmarked their results, which we consider promising.

There are many directions in which this work can be improved and extended to account for its limitations. The most obvious directions are by validating on more tasks or by accounting for more metrics in simulation such as different materials in compound objects, with different friction coefficients and softness values. This would also provide more realistic locations for the center of mass, which is crucial to determine the optimal grasp in many situations such as for the beating with hammers. As a natural follow up, we foresee a validation step on a real robotic arm to prove the generalization capability from simulation to reality. Indeed, we plan to integrate the learned models with a real Barrett hand and test on similar manipulators (with three or even two fingers) to assess the robustness of performance with inaccurate manipulator models. An important limitation of the current work is the unstructured definition of the affordance functions $\tilde{F}_T$ which at this stage *define* the tasks themselves for the system. A direction of improvement is towards a different description of tasks from which functions $\tilde{F}_T$ can be extracted or learned. Reference [25] extracts a representation of a task in terms of relevant physical quantities from the demonstration of a human choosing a tool and executing the task, such representation can be used to drive the automatic synthesis of the affordance function from a human demonstration. An alternative approach can be the definition of the execution of the task and a performance index of its end effectiveness: this would allow the optimization of the affordance function by reinforcement learning by simulation of the execution of the task itself with no further human intervention.

REFERENCES

[1] H. Dang and P. K. Allen, "Tactile experience-based robotic grasping," in *Workshop on Advances in Tactile Sensing and Touch based Human-Robot Interaction, HRI*, 2012.

[2] 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," *The International Journal of Robotics Research*, vol. 37, no. 4-5, pp. 421–436, 2018.

[3] D.-J. Kim, R. Lovelett, and A. Behal, "Eye-in-hand stereo visual servoing of an assistive robot arm in unstructured environments," in *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*. IEEE, 2009, pp. 2326–2331.

[4] A. N. Erkan, O. Kroemer, R. Detry, Y. Altun, J. Piater, and J. Peters, "Learning probabilistic discriminative models of grasp affordances under limited supervision," in *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*. IEEE, 2010, pp. 1586–1591.

[5] A. Zeng, S. Song, K.-T. Yu, E. Donlon, F. R. Hogan, M. Bauza, D. Ma, O. Taylor, M. Liu, E. Romo *et al.*, "Robotic pick-and-place of novel objects in clutter with multi-affordance grasping and cross-domain image matching," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 1–8.

[6] M. A. Roa and R. Suárez, "Grasp quality measures: review and performance," *Autonomous robots*, vol. 38, no. 1, pp. 65–88, 2015.

[7] A. T. Miller and P. K. Allen, "Examples of 3d grasp quality computations," in *Proceedings 1999 IEEE International Conference on Robotics and Automation (Cat. No. 99CH36288C)*, vol. 2. IEEE, 1999, pp. 1240–1246.

[8] W. Townsend, "Mcb-industrial robot feature article-barrett hand grasper," *Industrial Robot: An International Journal*, vol. 27, no. 3, pp. 181–188, 2000.

[9] A. T. Miller and P. K. Allen, "Graspit! a versatile simulator for robotic grasping," *IEEE Robotics Automation Magazine*, vol. 11, no. 4, pp. 110–122, Dec 2004.

[10] P. Shilane, P. Min, M. Kazhdan, and T. Funkhouser, "The princeton shape benchmark," in *Proceedings Shape Modeling Applications, 2004.* IEEE, 2004, pp. 167–178.

[11] J. Gibson, *The senses considered as perceptual systems*. Boston: Houghton Mifflin, 1966.

[12] C. Michaels, "Affordances: Four points of debate," *ECOLOGICAL PSYCHOLOGY*, vol. 15, pp. 135–148, 04 2003.

[13] E. Şahin, M. Çakmak, M. R. Doğar, E. Uğur, and G. Üçoluk, "To afford or not to afford: A new formalization of affordances toward affordance-based robot control," *Adaptive Behavior*, vol. 15, no. 4, pp. 447–472, 2007.

[14] M. Prats, P. J. Sanz, and A. P. Del Pobil, "Task-oriented grasping using hand preshapes and task frames," in *Robotics and Automation, 2007 IEEE International Conference on*. IEEE, 2007, pp. 1794–1799.

[15] D. Song, K. Huebner, V. Kyrki, and D. Kragic, "Learning task constraints for robot grasping using graphical models," in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2010, pp. 1579–1585.

[16] H. Dang and P. K. Allen, "Semantic grasping: Planning robotic grasps functionally suitable for an object manipulation task," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Oct 2012, pp. 1311–1317.

[17] H. Dang and P. K. Allen, "Semantic grasping: planning task-specific stable robotic grasps," *Autonomous Robots*, vol. 37, no. 3, pp. 301–316, 2014.

[18] 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)*. IEEE, 2017, pp. 3266–3273.

[19] S. R. Lakani, A. J. Rodríguez-Sánchez, and J. Piater, "Towards affordance detection for robot manipulation using affordance for parts and parts for affordance," *Autonomous Robots*, vol. 43, no. 5, pp. 1155–1172, 2019.

[20] A. Myers, C. L. Teo, C. Fermüller, and Y. Aloimonos, "Affordance detection of tool parts from geometric features," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2015, pp. 1374–1381.

[21] M. Ciocarlie, C. Goldfeder, and P. Allen, "Dexterous grasping via eigengrasps: A low-dimensional approach to a high-complexity problem," in *Robotics: Science and Systems Manipulation Workshop-Sensing and Adapting to the Real World*. Citeseer, 2007.

[22] 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.

[23] C. Cgal, "Computational geometry algorithms library," 2008.

[24] Q.-Y. Zhou, J. Park, and V. Koltun, "Open3D: A modern library for 3D data processing," *arXiv:1801.09847*, 2018.

[25] Y. Zhu, Y. Zhao, and S. Chun Zhu, "Understanding tools: Task-oriented object modeling, learning and recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 2855–2864.

Towards Affordance Prediction with Vision via Task Oriented Grasp Quality Metrics

Luca Cavalli, Gianpaolo Di Pietro, Matteo Matteucci
luca3.cavalli@polimi.it, gianpaolo.dipietro@polimi.it, matteo.matteucci@polimi.it

POLITECNICO MILANO 1863

2nd IWCMAR @ International Conference on Robotics and Automation

Proposed Task Oriented Grasping Framework

Define affordance with respect to a task as a function of basic grasp metrics encoding geometry and statics of any hypothesis on grasp and point of use.

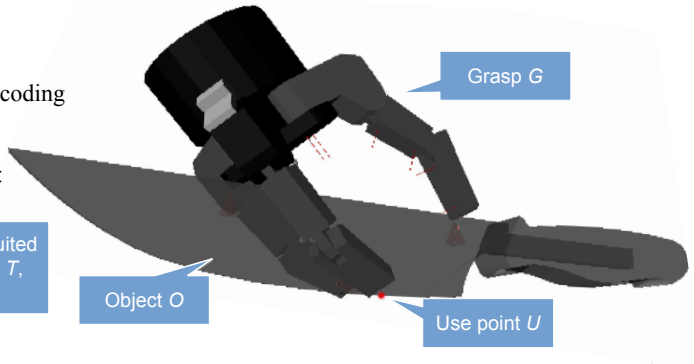
Affordance function for task *T* based on object *O*, grasp *G*, and usage point *U*:

$$F_T : (O, G, U) \mapsto \mathbb{R}$$

The higher the more suited (*O, G, U*) are for task *T*, e.g., for cutting

Approximated as a function of base grasp metrics:

$$\tilde{F}_T : \phi \in \mathbb{R}^n \mapsto \mathbb{R}$$



Base Grasp Metrics

We measure properties of the grasp by its geometry and statics

- Grasp robustness (ε ∈ R)
- Rotational inertia (I ∈ R)
- Hand effort on impact (Ei∈R)
- Hand effort on hold (Eh∈R^6)
- Momentum discharge efficiency (δ∈R)
- Force transmitted to use (Uτ∈R)
- Use local geometry (Ug∈R)

These should/could be learned instead of coded!

We hard code affordance function approximations for some sample tasks (picking, cutting, beating) via relevant basic grasp metrics

Input: E_h
Output: $\tilde{F}_{pick}$
1: **return** $-\sum_{i=1}^{6} E_h[i]$

Input: ϵ, U_τ, U_g
Output: $\tilde{F}_{cut}$
1: **if** $\epsilon < \tau_\epsilon$ OR $U_g < \tau_{U_g}$ **then**
2: **return** $-\infty$
3: **else**
4: **return** U_τ
5: **end if**

Input: $\epsilon, \delta, I, E_i, E_h$
Output: $\tilde{F}_{beat}$
1: **if** $\epsilon < \tau_\epsilon$ OR $\delta < \tau_\delta$ OR $\sum_{i=1}^{6} E_h[i] == \infty$ **then**
2: **return** $-\infty$
3: **else**
4: **return** $\frac{I}{E_i}$
5: **end if**

Metrics inference from vision

We infer metrics Φ from a depth image D of the object, an initial hand position p_0 and a use direction d assuming a fixed grasp policy GP and a direction to location mapping DM

$$\mathcal{M}^{\Phi} : (p_0, D(p_0, O), d) \mapsto \Phi(O, GP(p_0, O), DM(d, O))$$

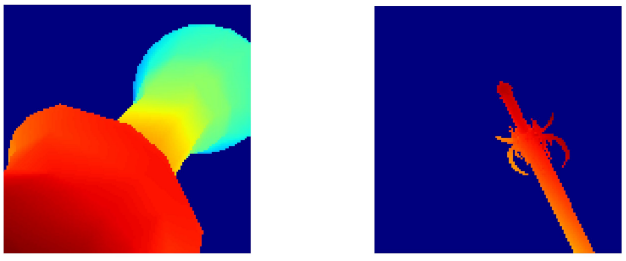


Fig. 9. Synthesized range images with camera-in-hand perspective.

Framework Validation

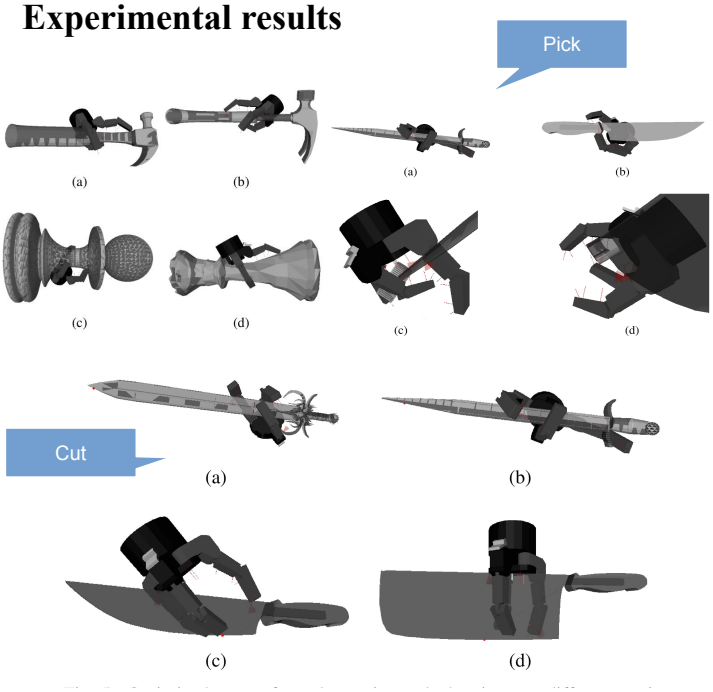
We built a scalable GraspIt! plugin to simulate random grasps and automate metric data collection

We collected a dataset of millions of grasps with pre-computed metrics by simulation via the GraspIt! simulator with the purpose of learning inference

We qualitatively show that maxima in the affordance function of some selected tasks correspond to semantically coherent grasps with interesting emerging behaviours

We assess the feasibility of learning a model that can infer basic grasp metrics from partial information about a target object

Experimental results



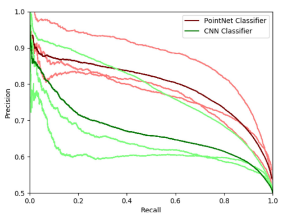


Fig. 10. Recall-precision curve for CNN and PointNet classifiers. Dark lines are the curve on all the objects of the test set for the respective model, light lines are computed on single objects of the same set

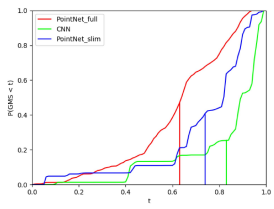


Fig. 11. Global minimum score cumulative distributions for the three regressors. The PointNet full regressor is the standard PointNet, the PointNet slim is the standard PointNet with transformation layers fixed to the identity. The vertical lines of each distribution is the expected value for the GMS.