Learning Grasping and Walking Motion Generation for Humanoid Robots

Dissertation

zur Erlangung des Doktorgrades (Dr. rer. nat.)
der Mathematisch-Naturwissenschaftlichen Fakulät
der Rheinischen Friedrich-Wilhelms-Universität Bonn

vorgelegt von

Diego Alexander Rodriguez Vargas
aus Bogotá, Kolumbien

Bonn, Oktober 2020

Angefertigt mit Genehmigung der Mathematisch-Naturwissenschaftlichen
Fakultät der Rheinischen Friedrich-Wilhelms-Universität Bonn

Erster Gutachter	Prof. Dr. Sven Behnke
Zweiter Gutachter	Prof. Dr. Tamin Asfour
Tag der Promotion	14. April 2021
Erscheinungsjahr	2021

Abstract

For acting in human-made scenarios, humanoid robots are undoubtedly the most versatile and flexible platforms among a vast number of available robotic systems. However, this versatility comes at the cost of complexity. Dexterous grasping and bipedal locomotion pose still several challenges in terms of planning and control mainly due to the high dimensionality, complex dynamics and real-time constraints. Inspired by human nature, learning approaches offer a promising alternative to address these issues. By leveraging on prior knowledge and experiences, represented as neural networks, latent spaces, probabilistic models, among others, this thesis present novel learning approaches to generate grasping and walking motions for humanoid robots.

Initially, geometrical variations of an object category are aggregated into a latent (shape) space in order to register novel object shapes in a non-rigid fashion. Grasping knowledge is then transferred to novel instances based on their object shape. This knowledge includes approaching motions and the joint configuration of multi-fingered robotic hands, whose inherent high-dimensionality is handled by learning postural synergies. The object registration can be performed online with 3D sensors or RGB cameras. The shape inference from RGB images is especially relevant to objects challenging to perceive by depth sensors, e.g., those with transparent or shiny surfaces. The proposed grasping approaches put particular emphasis on providing functional grasps that enable not only to pick objects but to use them. The grasping transfer is evaluated in several robotic platforms in single and dual arm applications.

On the second part of this thesis, the attention is confined to the optimization and generation of bipedal walking motions. By means of Gaussian processes, the dissimilarity between a physics-based simulator and a real humanoid robot is characterized. Thus, experiments performed in simulation and with the real platform are integrated into a sample-efficient Bayesian optimizer that selects the most informative parameters to evaluate dictated by the relative entropy of a cost function. Finally, the advantages and applicability of recent deep reinforcement learning methods on locomotion controllers are discussed. A novel approach is presented that learns a single control policy capable of omnidirectional walking without any analytical gait or any previous notion of walking. The robustness and omnidirectional capabilities of the learned walking controller is evaluated in a series of experiments and the learned gait is successfully transferred to the real hardware.

Zusammenfassung

Unter der großen Anzahl verfügbarer Robotersysteme sind Humanoide Roboter zweifellos die anpassungsfähigsten und flexibelsten Plattformen für das Agieren in menschengemachten Umgebungen. Diese Flexibilität geht jedoch mit höherer Komplexität einher. Geschicktes Greifen und bipedale Fortbewegung stellen immer noch große Herausforderungen in Bezug auf Planung und Steuerung dar, hauptsächlich aufgrund der hohen Dimensionalität, der komplexen Dynamik und der Echtzeitanforderungen. Inspiriert von der menschlichen Natur bieten Lernansätze eine vielversprechende Alternative zu klassischen Methoden, um diese Probleme anzugehen. In dieser Arbeit werden neue Lernansätze zur Erzeugung von Greif- und Gehbewegungen für humanoide Roboter vorgestellt, bei denen auf Vorwissen zurückgegriffen wird, welches unter anderem als neuronale Netze, latente Räume und Wahrscheinlichkeitsmodelle repräsentiert wird.

Zunächst werden geometrische Variationen einer Objektkategorie zu einem latenten Raum zusammengefasst, um die nicht-rigide Registrierung von neuartigen Objektformen durchzuführen. Gelernte Greiffähigkeiten werden dann basierend auf der Objektform auf neuartige Instanzen übertragen. Dieses Fähigkeiten umfassen sowohl Armbewegungen, um sich dem Objekt anzunähern als auch die Gelenkkonfiguration von Roboterhänden mit mehreren Fingern, deren inhärente Hochdimensionalität durch das Lernen von Haltungssynergien reduziert wird. Die Objektregistrierung kann online mit 3D-Sensoren oder RGB-Kameras durchgeführt werden. Die Forminferenz aus RGB-Bildern ist besonders relevant für Objekte, deren Wahrnehmung durch 3D-Sensoren herausfordernd ist, z.B. solche mit transparenten und glänzenden Oberflächen. Die vorgeschlagenen Ansätze zur Griffplanung legen besonderen Wert auf Funktionsgriffe, mit denen Objekte nicht nur gegriffen, sondern auch verwendet werden können. Die Griffübertragung wird auf verschiedenen ein- und zweiarmigen Roboterplattformen ausgewertet.

Im zweiten Teil dieser Arbeit wird der Fokus auf die Optimierung und das Lernen von bipedalen Gehbewegungen gelegt. Mittels Gaußscher Prozesse wird die Differenz zwischen einem physikbasierten Simulator und einem echten humanoiden Roboter beschrieben. Damit werden Experimente, die in der Simulation und mit der realen Plattform durchgeführt werden, in einem stichprobeneffizienten Bayes'schen Optimierer kombiniert, der die Parameter mit hohem Informationsgehalt anhand der relativen Entropie einer Kostenfunktion zur Bewertung auswählt. Abschließend wird die Anwendbarkeit der neuesten Lernmethoden des Bestärkenden Lernens auf das Problem der Fortbewegungssteuerung diskutiert. Ein neuartiger Ansatz zur omnidirektionalen Fortbewegung wird vorgestellt, der eine vollständige Kontrollstrategie lernt, ohne dass eine analytische

Beschreibung des Gangs vorliegt. Die Robustheit und die omnidirektionalen Fähigkeiten des erlernten Laufreglers werden in einer Reihe von Experimenten analysiert und der erlernte Gang wird erfolgreich auf reale Hardware übertragen.

Acknowledgments

First of all, I would like to thank Prof. Sven Behnke for giving me the opportunity to work in his group and for providing me with a nurturing environment to develop my ideas that resulted in the approaches presented in this thesis. His guidance and insightful discussions have played a key role for my research in the last years.

My deepest gratitude goes to my parents for their wholehearted support — especially in the hardest moments. I would like to thank my sibling for being unconditionally at my side without faltering. They have constantly bolstered my confidence in this challenging path. I extend my special gratitude to my friends for their ceaseless encouragement and patience.

I would like to express my gratitude to all current and previous members of the Autonomous Intelligent Systems group with whom I had the opportunity to work and to spend time together. In special, I thank Max Schwarz for insightful discussions and his willingness to support me in several software-related conundrums. For the helpful advice and assistance with mechatronic problems I thank Michael Schreiber, a person that support my work with creative and practical solutions. I am also grateful with the NimbRo soccer team for the fruitful and enjoyable time we spent before and during every RoboCup competition I participated.

This thesis would not have been possible without the hard work of the Autonomous Intelligent Systems students, especially, the work of Andre Brandenburger, Florian Huber and Corbin Cogswell.

This work was partially funded by the European Union's Horizon 2020 Programme under Grant Agreement 644839 (CENTAURO) and the German Research Foundation (DFG) under the grant BE 2556/13 and the grant BE 2556/12 ALROMA in priority programme SPP 1527 Autonomous Learning.

Contents

1 Introduction

"What we call the beginning is often the end. And to make an end is to make a beginning. The end is where we start from."

— T. S. Eliot.

How can humanoid robots learn to move in human-made scenarios? The complexity and dynamism of real world environments pose several challenges for achieving full robot autonomy and interaction. Despite the rapid advances in artificial intelligence and robotics in the last decade, the deployment of autonomous humanoid robots in society remains heretofore purely in science fiction stories.

The approaches presented in this thesis make a step forward to deploy humanoid robots in quotidian scenarios. As humans, robots face multiple challenges when interacting in their surroundings. However, in contrast to people, robots need to adapt to daily scenarios that are not designed neither by, nor for them. Having a similar morphology as the human one, enables humanoid robots to better interact in real world environments, where versatility is an imperative requirement considering the diversity of capabilities such scenarios demand.

In this thesis, two fundamental capabilities of humanoid robots are studied, namely *grasping* and *walking*. Specifically, the attention will be confined to the motion generation and optimization problems. Initially, efforts are concentrated to the upperbody of humanoid robots, where grasp motions are planned which consider robot arms and multi-fingered hands. Later, the center of interest is shifted to the whole body in order to optimize and to generate omnidirectional bipedal gaits.

A common challenge that emerges naturally in the motion generation of robotic systems is their high dimensionality. It often implies the control of a humanoid upperbody including fingered hands for grasping tasks or of the whole body for locomotion behaviors. This high dimensionality becomes even more relevant when it is combined with real time constraints, which impose limits on the computational requirements of the employed approaches.

Additional challenges to the deployment of autonomous robots in daily environments are the unpredictability, dynamism and variability of such scenarios. Thus, knowledge adaptation and generalization are not only desired but demanded to deal with the complexity of those potentially changing scenarios. Challenges are evidenced, for example, when robots are required to move on different surfaces or to grasp novel objects seen for the first time. Furthermore, full robot autonomy, i.e., the capacity of robots to operate without human supervision, requires a high level of system robustness and integration.

One specific difficulty associated to grasping lies on guarantying a *functional* usage of unknown objects after its grasp. In this manner, for instance, grasping a drill should enable its use to make holes. This functional usage demands object understanding by the robot, which is unfeasible to acquire online due to time constraints. The problem of generating functional grasps becomes harder in cluttered and unstructured scenarios, where objects are partially observed. In other words, robots need to be able to reconstruct the non-visible or occluded parts of the objects.

Regarding bipedal walking, the full robot dynamics is often simplified for planning and control tasks due to its complexity and computational cost, giving place to the incorporation of simple models such as the inverted pendulum and the centroidal dynamics. Moreover, limitations of the available sensing and actuation capabilities, such as sensor noise and actuation latencies, make difficult the deployment of locomotion controllers on real hardware.

Inspired by human nature, learning approaches are proposed throughout this thesis to address these difficulties. Different models are learned by labeled data, (e.g., Convolutional Neural Networks [CNNs] and Gaussian Processes [GPs]), unlabeled data (e.g, latent spaces), or by interacting with the environment by Reinforcement Learning (RL) methods. These learned models accumulate specific knowledge such as the object shape variability, and robot model dynamics, which are exploited online to meet real time constraints.

The work presented in this thesis is based on simple but powerful observations on how humans learn. For grasping, for instance, people are able to manipulate objects with a given degree of familiarity that are presented for the first time without training and often without mistakes. In the same manner, the core component of the grasping methods presented in this thesis, aggregates object knowledge into a latent space that models typical intra-class shape variability, which is used to transfer previously acquired grasping skills.

Another influential observation is the manner how toddlers learn to walk by trial and error based on a postural guide provided by the parents or by baby walkers without any detailed walking reference. Similarly, a novel approach to learn omnidirectional walking capabilities is proposed based on a postural guide (nominal pose) without a reference motion in a reinforcement learning framework.

The approaches developed in this work can be categorized under the area of robot learning applied to grasping and walking. Specifically, learning methods are developed on the fields of object (knowledge) representation, grasp planning, gait optimization and locomotion control.

1.1 Key Contributions

This thesis presents novel approaches for the generation of grasping and walking motions of humanoid robots. These methods allow such robots:

- to generate models of object shapes to register non-rigidly partially observed unknown objects;
- to transfer grasping skills to novel objects;
- to control multi-fingered hands for grasping tasks;
- to optimize gait parameters to increase robot stability, and;
- to learn omnidirectional walking capabilities.

In more detail, this thesis makes the following contributions:

- *Shape Space Registration*. A novel learned non-rigid registration approach is proposed based on a shape (latent) space of deformation fields. The shape space encodes typical intra-class geometrical object variations, thus, this method can be applied to interpolate between instances of a class of objects to create novel instances.
- *Shape Space Registration from RGB Images*. Objects difficult to perceive with depth sensors, e.g., those with transparent or shiny surfaces, are registered and reconstructed based on a single RGB image thanks to prior knowledge represented in deep neural networks.
- *Grasping Skill Transfer*. Associated grasping knowledge of a canonical object instance is transferred to novel instances based on their shape descriptor inferred by the latent space described above. The skill transfer allows successfully single and dual arm autonomous grasping with real robots.
- *Grasp Posture of Multi-fingered Hands* based on the geometry of the objects. Grasp poses are described through postural synergies which are inferred based on the object shape. The synergy space is learned by demonstration using a hand exoskeleton which allows to evaluate the quality of the grasps by force feedback.
- *Gait Parameter Optimization Combining Simulation and Real Robot Experiments*. The dissimilarity between simulation and a real humanoid robot is learned through a Bayesian Optimization approach that samples evaluation points according to the relative entropy, i.e., most informative points.
- *Learning Omnidirectional Gait*. Deep Reinforcement Learning (DRL) methods are employed to learn a locomotion controller with omnidirectional capabilities. The learning process does not require reference motions and the resulting learned controller is successfully transferred to a real humanoid robot.

1.2 Publications

Parts of this thesis have been published in journals and conference proceedings. The most relevant publications are presented below in chronological order:

- D. Rodriguez, C. Cogswell, S. Koo, and S. Behnke [2018a]. "Transferring grasping skills to novel instances by latent space non-rigid registration". In: *IEEE International Conference on Robotics and Automation (ICRA)*
- D. Rodriguez and S. Behnke [2018]. "Transferring category-based functional grasping skills by latent space non-rigid registration". In: *IEEE Robotics and Automation Letters (RA-L)*
- D. Rodriguez, A. Di Guardo, A. Frisoli, and S. Behnke [2018c]. "Learning postural synergies for categorical grasping through shape space registration". In: *IEEE-RAS 18th International Conference on Humanoid Robots (Humanoids)*
- D. Rodriguez, A. Brandenburger, and S. Behnke [2018b]. "Combining simulations and real-robot experiments for Bayesian optimization of bipedal gait stabilization". In: *Proceedings of 22nd RoboCup International Symposium, Montreal, Canada*
- D. Rodriguez, F. Huber, and S. Behnke [2020]. "Category-level 3D non-rigid registration from single-view RGB images". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*
- D. Rodriguez and S. Behnke [2021]. "DeepWalk: Omnidirectional bipedal gait by deep reinforcement learning". In: *IEEE International Conference on Robotics and Automation (ICRA), to appear*

Sections of the following publications (organized in chronological order) are also presented in this thesis:

- D. Pavlichenko, D. Rodriguez, M. Schwarz, C. Lenz, A. S. Periyasamy, and S. Behnke [2018]. "Autonomous dual-arm manipulation of familiar objects". In: *IEEE-RAS 18th International Conference on Humanoid Robots (Humanoids)*
- T. Klamt, D. Rodriguez, M. Schwarz, C. Lenz, D. Pavlichenko, D. Droeschel, and S. Behnke [2018]. "Supervised autonomous locomotion and manipulation for disaster response with a centaur-like robot". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*
- T. Klamt, M. Schwarz, C. Lenz, L. Baccelliere, D. Buongiorno, T. Cichon, A. DiGuardo, D. Droeschel, M. Gabardi, M. Kamedula, N. Kashiri, A. Laurenzi, D.

Leonardis, L. Muratore, D. Pavlichenko, A. Periyasamy, D. Rodriguez, M. Solazzi, A. Frisoli, M. Gustmann, J. Rossmann, U. Suess, N. Tsagarakis, and S. Behnke [2019b]. “Remote mobile manipulation with the centauro robot: Full-body telepresence and autonomous operator assistance”. In: *Journal of Field Robotics (JFR)*

- D. Pavlichenko, D. Rodriguez, C. Lenz, M. Schwarz, and S. Behnke [2019]. “Autonomous bimanual functional regrasping of novel object class instances”. In: *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*
- T. Klamt, D. Rodriguez, L. Baccelliere, X. Chen, D. Chiaradia, T. Cichon, M. Gabardi, P. Guria, K. Holmquist, M. Kamedula, H. Karaoguz, N. Kashiri, A. Laurenzi, C. Lenz, D. Leonardis, E. Mingo, L. Muratore, D. Pavlichenko, F. Porcini, Z. Ren, F. Schilling, M. Schwarz, M. Solazzi, F. Michael, A. Frisoli, M. Gustmann, P. Jensfelt, K. Nordberg, J. Rossmann, U. Suess, N. Tsagarakis, and S. Behnke [2019a]. “Flexible disaster response of tomorrow - Final presentation and evaluation of the CENTAURO system”. In: *IEEE Robotics and Automation Magazine (RAM), Special Issue on Humanoid Robot Applications in Real World Scenarios*

The following publications are closely related to the topics presented in this thesis and were written during the time in which the presented research has been conducted.

- G. Ficht, H. Farazi, A. Brandenburger, D. Rodriguez, D. Pavlichenko, P. Allgeuer, M. Hosseini, and S. Behnke [2018]. “NimbRo-OP2X: Adult-sized open-source 3D printed humanoid robot”. In: *IEEE-RAS 18th International Conference on Humanoid Robots (Humanoids)*
- G. Ficht, H. Farazi, D. Rodriguez, D. Pavlichenko, P. Allgeuer, A. Brandenburger, and S. Behnke [2020]. “NimbRo-OP2X: Affordable adult-sized 3D-printed open-source humanoid robot for research”. In: *International Journal of Humanoid Robotics (IJHR)*

1.3 Outline

This thesis is composed by eight Chapters. The scientific contributions are presented from Chapter 2 to Chapter 7, which are organized thematically in two Parts: Grasping (Part. I) and Walking (Part. II). Each of the content chapters starts with the problem formulation followed by a comparative analysis of the relevant state of the art. Each content chapter finalizes with the evaluation of the proposed methods and a discussion of the advantages, limitations, and future work. Approaches presented in Chapters 3, 4, and 5 build upon the content of Chapter 2.

Chapter 2 presents the shape space non-rigid registration approach. The theoretical foundation for several key concepts in this thesis such as deformation fields and the construction of latent spaces is presented. It is described how the objects are abstracted and reconstructed from partially observed data. The method is evaluated against typical error sources in real applications such as sensor noise and pose misalignments. Moreover, the applicability of the approach in online tasks is demonstrated.

In *Chapter 3*, objects are registered non-rigidly based on a single observed RGB image. In this manner, a canonical model can be deformed towards observed objects that are difficult to perceive by 3D sensors, e.g., those with transparent surfaces. A CNN is trained to infer deformation vectors of the visible parts of the novel objects. Shape reconstruction is provided by employing a shape space which additionally infers the deformation of the non-visible parts. The method is evaluated in different object categories with synthetic and real images.

In *Chapter 4*, the shape registration method introduced in Chapter 2 is employed to transfer grasping skills. Control poses of a grasping motion of a canonical model are warped according to the shape dissimilarities observed in the observed object. In addition, knowledge of different successful grasps are aggregated into a learned model. The transfer is evaluated in single and dual-arm platforms where a direct functional grasp is feasible, and in regrasping experiments where an initial grasp needs to be executed first to get control over the object pose.

Chapter 5 addresses the control of multi-fingered hands based on the object's geometry. Grasp hand poses are represented as postural synergies in order to reduce the higher dimensionality of the studied robotic hand. A synergy space is constructed by grasping representative objects by means of an exoskeleton which allows to assess the grasp quality by force feedback. The object's geometry is represented as a shape descriptor given by the method developed in Chapter 2. The developed approach is evaluated in simulation and with real robot experiments with different object categories.

In the second Part of this thesis, learning approaches are examined for walking. Initially, in *Chapter 6*, control parameters are optimized to increase the gait stability of a humanoid robot. The number of real-world samples and the corresponding wear-off of the robot platform is reduced by means of simulated walking sequences. The integration of simulated experiments is possible by modeling the sim-to-real error into the optimization problem. Candidates points to evaluate are selected according to their relative entropy, in other words, the most informative points are evaluated. In the experimental section it is shown how the walking stability of a humanoid robot is increased over a series of walking sequences.

Chapter 7 studied the generation of gait patterns from a learning perspective. A novel approach is presented that learns a locomotion controller with omnidirectional capabilities. A reinforcement learning problem is formulated that guides the policy search without reference motions and considering actuator limitations. The approach addresses the difficult challenges to transfer the learned gait to the real hardware. The learned locomotion controller is evaluated and successfully transferred to the real robot platform.

Finally, this thesis is concluded in *Chapter 8*. The accomplished scientific results are discussed putting in context their impact and limitations. Future research directions are also examined.

Each of the content chapters has an accompanying video that briefly presents the method and shows relevant media material such as animations and video footage of performed real robot experiments. The reader is encouraged to take a look into the videos while reading the corresponding chapters. All videos are available online [1].

Furthermore, please note that, in the Bibliography, direct links to online versions of the publications that are publicly available are provided in order to facilitate literature review of the reader.

1.4 Open Source Software Releases

An implementation of the shape space registration approach described in Chapter 2 is provided [2]. The current implementation learns shape spaces given a number of instances belonging to an object category and it is able to deform a chosen canonical instance into novel observed category instances. The code is integrated with the Robot Operating System (ROS). This release gives other researchers the opportunity to use and to extend the proposed approaches in their own research directions. In addition, it facilitates the comparison and validation of the reported results. The main features of the code release can be seen in Video 1.1 [3].

In addition, the code to generate geometrical primitives such as spheres, boxes, cylinders and capsules of robot meshes is also released[4]. The software outputs directly the robot description defined by an URDF file. The use of primitives plays a key role in the simulation of grasping and walking tasks, since the computation cost of collision checking queries is drastically reduced.

[1] Accompanying video material: `https://zenodo.org/record/3991986`

[2] `https://github.com/AIS-Bonn/shape_registration.git`

[3] Video 1.1: `https://zenodo.org/record/3991986/files/shape_space_code_release.mp4`

[4] `https://github.com/AIS-Bonn/primitive_fitter.git`

Part I

Grasping

2 Category-Level 3D Non-Rigid Registration from 3D Sensor Data

"The artist must have something to say, for mastery over form is not his goal but rather the adapting of form to its inner meaning."
— Wassily Kandinsky

People are able to dexterously manipulate objects even if they are presented by the first time. For instance, grasping a novel mug, pencil or screw driver. Grasping such objects happens effortless in humans because of all previous experiences with similar objects, or in other words, objects that belong to the same *category*. The body motions required to grasp and to manipulate such objects are adapted to the novel observed object geometry. Motions to grasp a wide mug are different than those for grasping a thinner one. The approaches presented in this thesis have been largely inspired by this observation. We aim to provide robots the capability of transferring knowledge between object instances based on their geometry. To achieve this, we propose a novel approach that is able to register non-rigidly object instances inside a category. Once the variability between instances is established, we can transfer knowledge between them. This chapter describes formally the object shape representation and registration, while Chapter 4 and Chapter 5 present the manipulation and grasping skill transfer process.

In this work, object shapes in a category are abstracted into a lower dimensional space, referred as *shape space*. Typical category-level geometrical variations are described by this shape space which allows to generate new instances by interpolation and extrapolation. Furthermore, using the shape space we can reconstruct partially observed objects (Figure 2.1) — desirable for grasp planning. This is especially relevant for grasping in unknown and unstructured environments where 3D models are not available and objects can not be fully observable directly, e.g., an object lying at the inner corner of a shelf.

During inference, the approach proposed here finds a transformation (expressed as a deformation field and a rigid transformation) from the canonical model to the novel instance. This is done by searching in the latent space— by linearly interpolating and

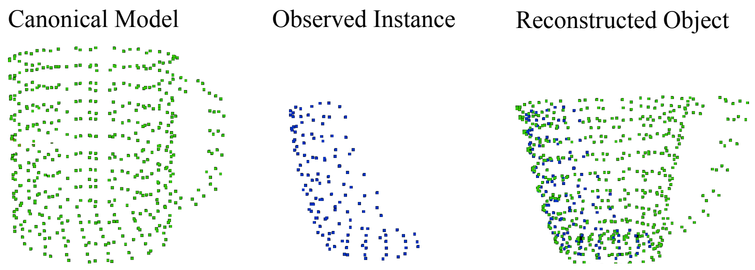

Figure 2.1: Non-rigid registration of a canonical model (green points) towards a partially observed instance (blue points). Thanks to the intra-class information encoded in the shape space, our approach is able to reconstruct the observed instance inferring missing parts such as the handle. The result of the registration is shown on the right.

extrapolating from other transformations found within the class— for the shape that best matches the observed 3D point set. In this manner, the shape parameters of the novel instance are estimated and occluded parts are inferred.

The approach described in this chapter has been published in [Rodriguez et al., 2018a] and [Rodriguez and Behnke, 2018]. The accompanying online Video 2.1[1] gives an overview of the approach presented in this Chapter. The video also includes visualization of the non-rigid registration of novel instances performed in real time.

2.1 Related Work

In contrast to rigid registration, where the problem is stated as finding a relative small number of parameters that fully define a rigid transformation between two feature sets, the non-rigid registration problem aims to deform a feature set, e.g., a point set, to match another feature set. This is a challenging task because the true underlying non-rigid transformations are often unknown. The non-rigid registration task has been addressed by imposing restrictions on the motions or deformations of the moving instance feature set. Different restrictions include: thin-plate splines [Chui and Rangarajan, 2003; Zou et al., 2007], isometry [Ovsjanikov et al., 2010; Tevs et al., 2009], Gaussian fields [Hahnel et al., 2003], conformal maps [Kim et al., 2011; Zeng et al., 2010], and motion coherence theory [Myronenko and Song, 2010].

In the recent past, several approaches have addressed the non-rigid registration as a probability density estimation problem [Horaud et al., 2010; Ma et al., 2016; Myronenko and Song, 2010]. This is done by estimating the parameters of Gaussian Mixture Models (GMMs) and by optimizing the maximum likelihood of points being drawn from theses GMMs by means of the Expectation Maximization (EM) algorithm [Dempster et al., 1977]. This idea can be extended in several manners. For example, Ma et al. [2016] incorporate

[1] Video 2.1: `https://zenodo.org/record/3991986/files/chapter_2.mp4`

local features such as shape context in the assignment of membership probabilities to preserve global and local structures. This provides robustness against noise, occlusions and/or outliers.

So far, all the mentioned approaches, however, do not explicitly model any prior knowledge about the morphology of the registered objects, thus, these methods do not perform well in general with partially observed data. The use of shape priors and lower dimensional latent spaces for non-rigid registration is a promising alternative to address this issue. Latent spaces have been proposed previously for body shapes [Allen et al., 2003; Hasler et al., 2009], brain images [Marsland et al., 2003], faces [Blanz and Vetter, 1999], and collections of shapes [Huang et al., 2012; Nguyen et al., 2011]. Blanz and Vetter [1999] created a morphable model of faces able to create novel faces and to interpolate between them. Similarly, Allen et al. [2003] created a shape space of human bodies using human body range scans with sparse 3D markers. Hasler et al. [2009] extended this space to include pose, creating a unified space of both pose and body shape. This allows them to model the surface of a body in various articulated poses more accurately. Nguyen et al. [2011] and Huang et al. [2012] established shape correspondences by creating collections of similar shapes and optimizing the mapping at a global scale.

The idea of a shape space has also been applied for dense shape reconstruction. Burghard et al. [2013] proposed an approach to estimate dense correspondences on a shape given initial coarse correspondences. They use the idea of minimum description length to create a compact shape space of related shapes with strongly varying geometry. Engelmann et al. [2016] learned a compact shape manifold which represents intra-class shape variance, allowing them to infer shape in occluded or noisy regions such as on textureless, reflective, or transparent surfaces. Dame et al. [2013] generate a shape space from TSDF (Truncated Signed Distance Function) to reconstruct the pose and 3D object shape, such that inferences can be incorporated into a monocular dense SLAM (Simultaneous Localization And Mapping) system. High level features from CAD models have been also used to perform the 3D reconstruction [Zia et al., 2013]. All these approaches, however, do not provide correspondences between points or do not offer any kind of transformation between the point sets, which limits its applicability to the skill transfer problem, where we need to deform not only object points, but also control poses and similar information.

2.2 Background

2.2.1 Coherent Point Drift

The category-level registration approach described in this chapter is based on the Coherent Point Drift (CPD) [Myronenko and Song, 2010], therefore the CPD method is briefly reviewed in this section.

For two known point sets: a template $\mathbf{S}^{[t]} = (\mathbf{s}_1^{[t]}, ..., \mathbf{s}_M^{[t]})^T \in \mathbb{R}^{M\times D}$ and a reference point set $\mathbf{S}^{[r]} = (\mathbf{s}_1^{[r]}, ..., \mathbf{s}_N^{[r]})^T \in \mathbb{R}^{N\times D}$, CPD calculates a deformation field that warps the template point set into the reference constrained by motion coherence. The non-rigid registration process is modeled by means of GMMs, where the points in $\mathbf{S}^{[t]}$ represent the centroids of the GMMs from which the points in $\mathbf{S}^{[r]}$ are drawn. In general, CPD optimizes the likelihood of the GMM and simultaneously imposes constraints on the motion of the centroids such that they move in a coherent manner, i.e., that points close to each other move similar to their neighbors [Yuille and Grzywacz, 1989].

For mathematical convenience, the negative log-likelihood function is minimized instead of directly maximizing the likelihood of the GMM:

$$E(\boldsymbol{\psi}, \sigma^2) = -\sum_{n=1}^{N} \log \sum_{m=1}^{M} \exp^{-\frac{1}{2\sigma^2}\left\|\mathbf{s}_n^{[r]} - \mathcal{T}(\mathbf{s}_m^{[t]}, v)\right\|^2}, \tag{2.1}$$

where $\mathcal{T}(\mathbf{s}_m^{[t]}, v)$ is a transformation from the template to the reference point set parameterized by the function v, and σ^2 is the isotropic covariance of the GMMs. CPD can be also applied to solve rigid registrations. For the non-rigid case, the transformation $\mathcal{T}$, is described by the original point set plus a displacement function v:

$$\mathcal{T}(\mathbf{S}^{[t]}, v) = \mathbf{S}^{[t]} + v(\mathbf{S}^{[t]}). \tag{2.2}$$

v defines how the points move. Thus, the regularization on the motion of the centroids is carried out directly on v. A regularization term $\phi(v)$, is consequently added to the negative log-likelihood Eq. (2.1):

$$f(v, \sigma^2) = E(\sigma^2, v) + \frac{\lambda}{2}\phi(v), \tag{2.3}$$

where the λ parameter trades off the motion regularization and the quality of the maximum likelihood fit. A particular choice of $\phi(v)$ leads to the following displacement function $v(\mathbf{Z})$ [Myronenko and Song, 2010]:

$$v(\mathbf{Z}) = G(\mathbf{S}^{[\mathbf{t}]}, \mathbf{Z})\mathbf{W}, \tag{2.4}$$

for a D-dimensional point set $\mathbf{Z}_{N\times D}$. The Gaussian kernel matrix $G(\mathbf{S}^{[\mathbf{t}]}, \mathbf{Z})$ is defined element-wise by:

$$g_{ij} = G(\mathbf{s}_i^{[t]}, \mathbf{z}_j) = \exp^{-\frac{1}{2\beta^2}\left\|\mathbf{s}_i^{[t]} - \mathbf{z}_j\right\|^2}. \tag{2.5}$$

$\mathbf{W}_{M\times D}$ is a set of D-dimensional deformation vectors, associated with each of the M points of $\mathbf{S}^{[\mathbf{t}]}$. $\mathbf{W}$ can be also interpreted as a matrix of kernel weights. The strength of the interaction between the points for constructing the Gaussian kernel matrix is controlled by the scalar β. For clarity in the notation, $\mathbf{G} \in \mathbb{R}^{M \times M}$, will refer to $G(\mathbf{S}^{[t]}, \mathbf{S}^{[t]})$.

Equation (2.3) is minimized by the Expectation Maximization (EM) algorithm. The Gaussian kernel matrix is calculated following Eq. (2.5), and the $\mathbf{W}$ matrix is initialized with zeros. In the expectation step, the posterior probabilities matrix $\mathbf{P}$ is computed, which is defined element-wise by:

$$p_{mn} = \frac{e^{-\frac{1}{2\sigma^2}\left\|\mathbf{s}_n^{[r]} - (\mathbf{s}_m^{[t]} + G(m,\cdot)\mathbf{W})\right\|^2}}{\sum_{m=1}^{M} e^{-\frac{1}{2\sigma^2}\left\|\mathbf{s}_n^{[r]} - (\mathbf{s}_m^{[t]} + G(k,\cdot)\mathbf{W})\right\|^2} + \frac{\omega}{1-\omega}\frac{(2\pi\sigma^2)^{\frac{D}{2}}}{N}}, \tag{2.6}$$

where the amount of noise is reflected by ω.

In the maximization step, the matrix $\mathbf{W}$ is estimated by:

$$(\mathbf{G} + \lambda\sigma^2 d(\mathbf{P1})^{-1})\mathbf{W} = d(\mathbf{P1})^{-1}\mathbf{P}\mathbf{S}^{[r]} - \mathbf{S}^{[t]}, \tag{2.7}$$

where $d(\cdot)^{-1}$ is the inverse diagonal matrix, and $\mathbf{1}$ describes a column vector of ones. The motivation and derivation of Eq. (2.7) as well as a deeper discussion of the CPD algorithm can be found in the original paper [Myronenko and Song, 2010].

2.3 Shape Space

The complete geometry or shape description of an object normally lies in very large dimensional spaces, e.g., the cardinality of point sets, which are expensive to manipulate. When the registration is performed between objects of the same category, typical geometrical variations can be used to enrich the registration process against occluded parts and to reduce the dimensionality of the object representation. This is possible by modeling common object shapes features. An object shape is then described as a combination of these shared common features. The manifold that contains these features is called *shape space* which is explained below.

An object category is defined as a set of objects with similar topology and extrinsic shape. Furthermore, for each category, a canonical model is chosen. The shape space is defined as a lower dimensional representation of the geometrical variations of the canonical model towards all the other object instances. In other words, an object instance will be then represented as the canonical model, which is shared across all other instances, and the abstract object features that explains the typical variations from the canonical model to the observed instance.

An object shape is represented as a point set. For point clouds, the positional information is used directly. For meshes, the underlying vertices can define the point set. If the point set is sparse, or it contains large amount of points, the underlying object points need to be filtered first, e.g., by grid filters or decimation operations. In case mesh manipulation is not desired, the point set can be generated by ray-casting the 3D mesh from several viewpoints on a tessellated sphere and by down-sampling with a voxel grid filter.

The canonical model is chosen by experts such that it represents a standard shape of the category. In order to guarantee consistence across all the object, all instances are aligned to a canonical frame. For instance, for *Mugs*, we can align the axis of the cylindrical part with the z axis of the canonical frame and vector from the axis of the cylinder to the intersection of the handle with the y axis.

The point set of the canonical model is defined as $\mathbf{C}$, which is deformed to match the point set of all other instances $\mathbf{T}_i$. The deformations are performed by CPD following Eq. (2.2) and Eq. (2.4):

$$\mathcal{T}_i(\mathbf{C}, \mathbf{W}_i) = \mathbf{C} + \mathbf{G}\mathbf{W}_i, \tag{2.8}$$

where $\mathbf{W}_i \in \mathbb{R}^{M \times D}$ is the deformation field computed by taking training example $\mathbf{T}_i$ as the reference point set $\mathbf{S}^{[\mathbf{r}]}$.

Observe from Eq. (2.8) that the registration from the canonical model to an observed instance depends only on $\mathbf{W}_i$, because $\mathbf{G}$ only requires $\mathbf{C}$ and it remains constant for all object instances. Thus, the deformation field for each instance $\mathbf{T}_i$ is fully captured by its matrix $\mathbf{W}_i$. Note, however, that vectors in $\mathbf{W}_i$ do not directly correspond with displacements, because a vector in $\mathbf{W}_i$ affects simultaneously several points according to the coherence matrix $\mathbf{G}$.

Moreover, observe that the dimensionality of $\mathbf{W}_i$ is the same across all instances. In addition, each row in $\mathbf{W}_i$ corresponds to a deformation vector of a specific point of $\mathbf{C}$. For example, if the m-row of $\mathbf{C}$ lies on a corner of a mug handle, all m-rows in the $\mathbf{W}_i$ matrices will describe the deformation of points around that corner of the canonical model to match the observed instances. These last two observations allow us to easily construct a latent lower-dimensional space.

To build the shape space of an object category, the $\mathbf{W}_i$ matrices are expressed as vectors $\mathbf{w}_i \in \mathbb{R}^{M \cdot D}$. The vectors are normalized to have zero-mean and unit-variance and are then assembled into a design matrix $\mathbf{Y}$. Finally, a lower-dimensional manifold of deformation fields is found by applying the Principle Component Analysis Expectation Maximization (PCA-EM) algorithm [Tipping and Bishop, 1999] on $\mathbf{Y}$. The expectation step is formulated as:

$$\mathbf{X} = \mathbf{Y}\mathbf{L}^T(\mathbf{L}\mathbf{L}^T)^{-1}, \tag{2.9}$$

and the maximization step as:

$$\mathbf{L} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{Y}. \tag{2.10}$$

$\mathbf{L} \in \mathbb{R}^{M \cdot D \times l}$ is the mapping matrix responsible for the dimensionality reduction. In this manner, any point $\mathbf{x} \in \mathbb{R}^l$ in the shape space can be transformed into a deformation field vector by $\mathbf{L}\mathbf{x}$ followed by the corresponding denormalization. Thus, moving through the l-dimensional space is analogous to linearly interpolating between the deformation

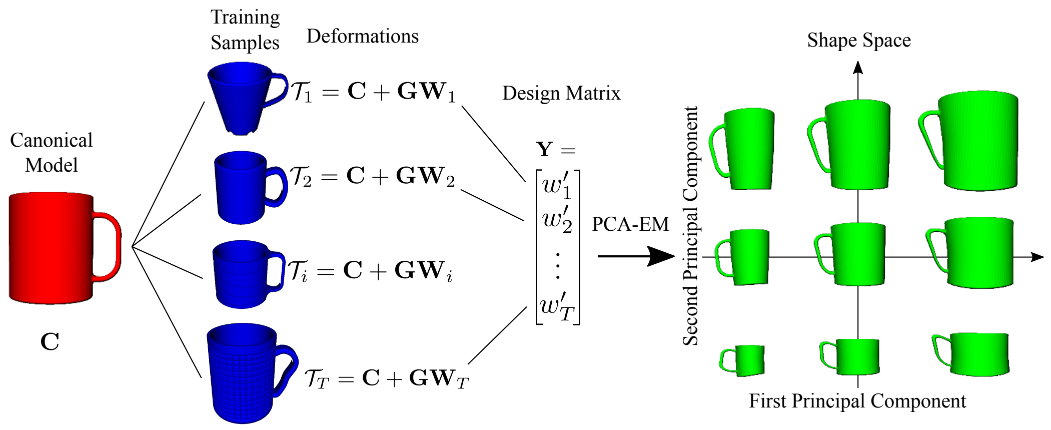
Training Samples
Deformations
Canonical Model
C
$\mathcal{T}_1 = \mathbf{C} + \mathbf{G}\mathbf{W}_1$
$\mathcal{T}_2 = \mathbf{C} + \mathbf{G}\mathbf{W}_2$
$\mathcal{T}_i = \mathbf{C} + \mathbf{G}\mathbf{W}_i$
$\mathcal{T}_T = \mathbf{C} + \mathbf{G}\mathbf{W}_T$
Design Matrix
$\mathbf{Y} = [w'_1\ w'_2\ \vdots\ w'_T]$
PCA-EM
Shape Space
Second Principal Component
First Principal Component

Figure 2.2: Building the shape space. The canonical model $\mathbf{C}$ is deformed to match all instances $\mathbf{T}_i$ using CPD. The deformations fields are assembled into a design matrix $\mathbf{Y}$. The shape space is the lower dimensional space spanned by the principal components of $\mathbf{Y}$.

fields. Similarly, an observation $\mathbf{w}$ is represented in the latent space by normalizing this vector and postmultyplying the result by $\mathbf{L}$. Thus, a high-dimensional shape represented by $M \cdot D$ parameters is now described by only l values.

A schematic overview of building the shape space is shown in Fig. 2.2.

2.4 Non-rigid Registration

In this section, the non-rigid registration process by means of a shape space is explained. As result of the registration, a shape space descriptor $\mathbf{x}$ and a rigid transformation $\boldsymbol{\theta}$ are delivered. The former describes the object shape by a latent vector $\mathbf{x}$, whereas the latter accounts for minor misalignments between the observed and the canonical shape at the global level.

The registration is formulated as a non-linear optimization problem. The optimization concurrently search for a shape descriptor $\mathbf{x}$ and a rigid transformation $\boldsymbol{\theta}$ in a gradient descent fashion. Thus, an aligned dense deformation field is found which when applied to the canonical shape $\mathbf{C}$, minimizes the distance to corresponding points of the observed shape $\mathbf{T}$. For the optimization problem, the following objective function is proposed:

$$E(\mathbf{x}, \boldsymbol{\theta}) = \sum_{m=1}^{M} \min_{n} \|\mathbf{T}_n - \Theta(\mathcal{T}_m(\mathbf{C}_m, \mathbf{W}_m(\mathbf{x})), \boldsymbol{\theta})\|^2, \tag{2.11}$$

that is simply the distance to the closest point. Observe that the objective function is non-linear because of the rotation in θ. The non-linear optimization was implemented using the Ceres solver [Agarwal et al., 2012].

When the optima $\mathbf{x}^*$ and $\boldsymbol{\theta}^*$ are found, and consequently $\mathbf{W}^*$, any point or set of points can be transformed into the manifold of the observed instance by using Eq. (2.4) and Eq. (2.2) and by applying the rigid transformation $\boldsymbol{\theta}^*$. Note, that as result, the non-rigid registration provides a dense deformation field, which allows to find deformation vectors for novel points, even for those added after the field is calculated. A j-vector of $\mathbf{W}^*$ influences several points of the canonical model that lie around the point j of $\mathbf{C}$. In this manner, vectors of $\mathbf{W}^*$ can be interpreted as influence regions mainly defined by β. When considering a novel point $\mathbf{z}$ that does not belong to the canonical model, by computing $G(\mathbf{z}, \mathbf{C})$, weights to each of the canonical points are computed, or, in other words, the weights of these influence regions are calculated. The deformed $\mathbf{z}$ is then expressed as:

$$\mathbf{z}' = \mathbf{z} + G(\mathbf{z}, \mathbf{C})\mathbf{W}^* \tag{2.12}$$

2.5 Evaluation

The shape space registration method proposed in this thesis has been tested in the following categories: *Mug*, *Drill*, *Spray bottle*, *Camera*, *(Drinking) Bottle*, *Watering can* and *Glass*. The models were obtained from online databases: Sketchfab[2], GrabCad[3], 3DWarehouse[4], and from the database recently released by Wang et al. [2019]. Samples of the 3D models are shown in Fig. 2.3.

For the evaluation, the point sets were established by ray-casting the 3D models from several viewpoints on a tessellated sphere and by down-sampling with a voxel grid filter. The shape spaces were built with the following number of instances for each category: *Mug* (21), *Drill* (16), *Spray bottle* (12), *Camera* (15), *Bottle* (14), *Watering can* (8), and *Glass* (16). Qualitative results of the registration are shown in Fig. 2.4, where the canonical models of each of the mentioned categories are deformed towards single partial views of the observed instances. Note how the handle of the mug is inferred as result of the non-rigid registration thanks to the knowledge represented in the shape space. Moreover, the physical dimensions of the canonical models varies a lot with respect to the observed instance. This demonstrates the large deformations (enlargements and reductions) that the registration is able to handle successfully. Furthermore, the robustness of the registration against misalignments is exhibited especially on the camera, bottle and watering can instances. This highlights the importance of the rigid transformation incorporated in the non-linear optimization in Eq. (2.11).

[2] https://sketchfab.com/

[3] https://grabcad.com/library

[4] https://3dwarehouse.sketchup.com/

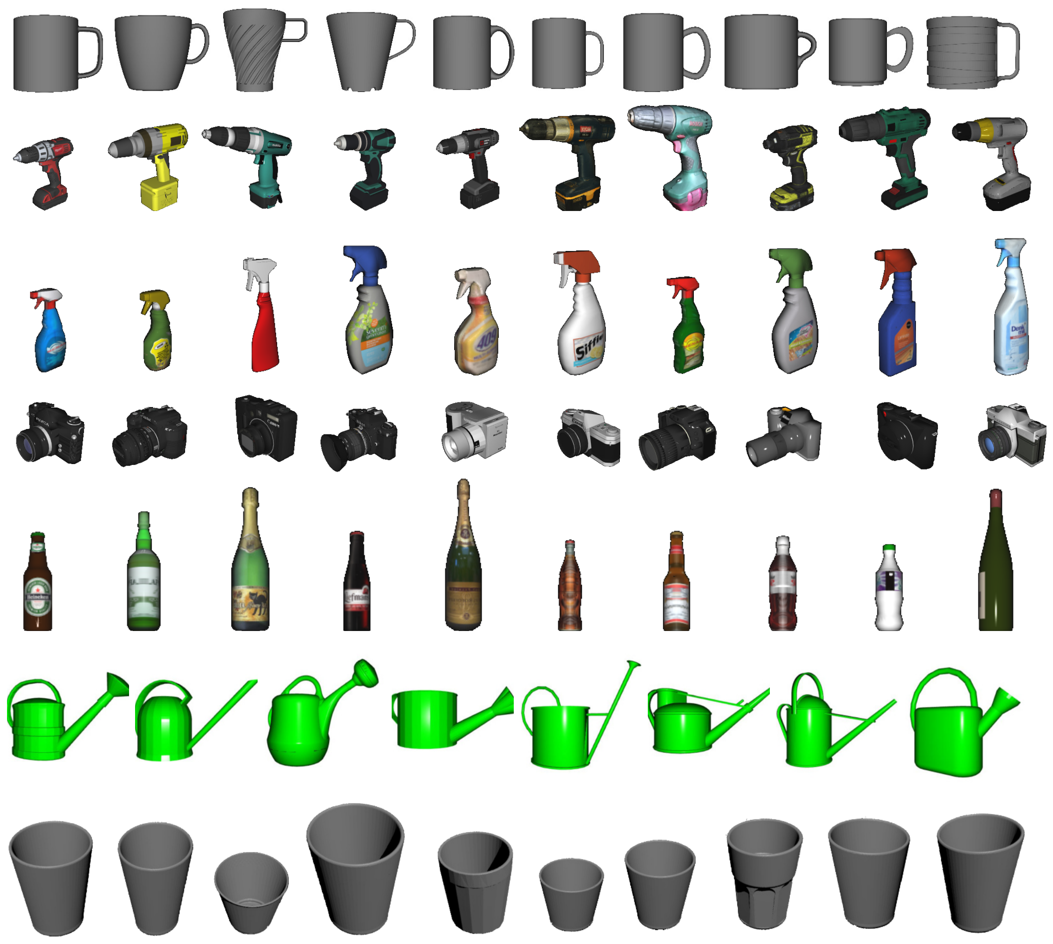
Figure 2.3: Samples of 3D models of the categories: *Mugs*, *Drill*, *Spray bottle*, *Camera*, *(Drinking) Bottle*, *Watering can* and *Glass*. The canonical models are presented at the leftmost column.

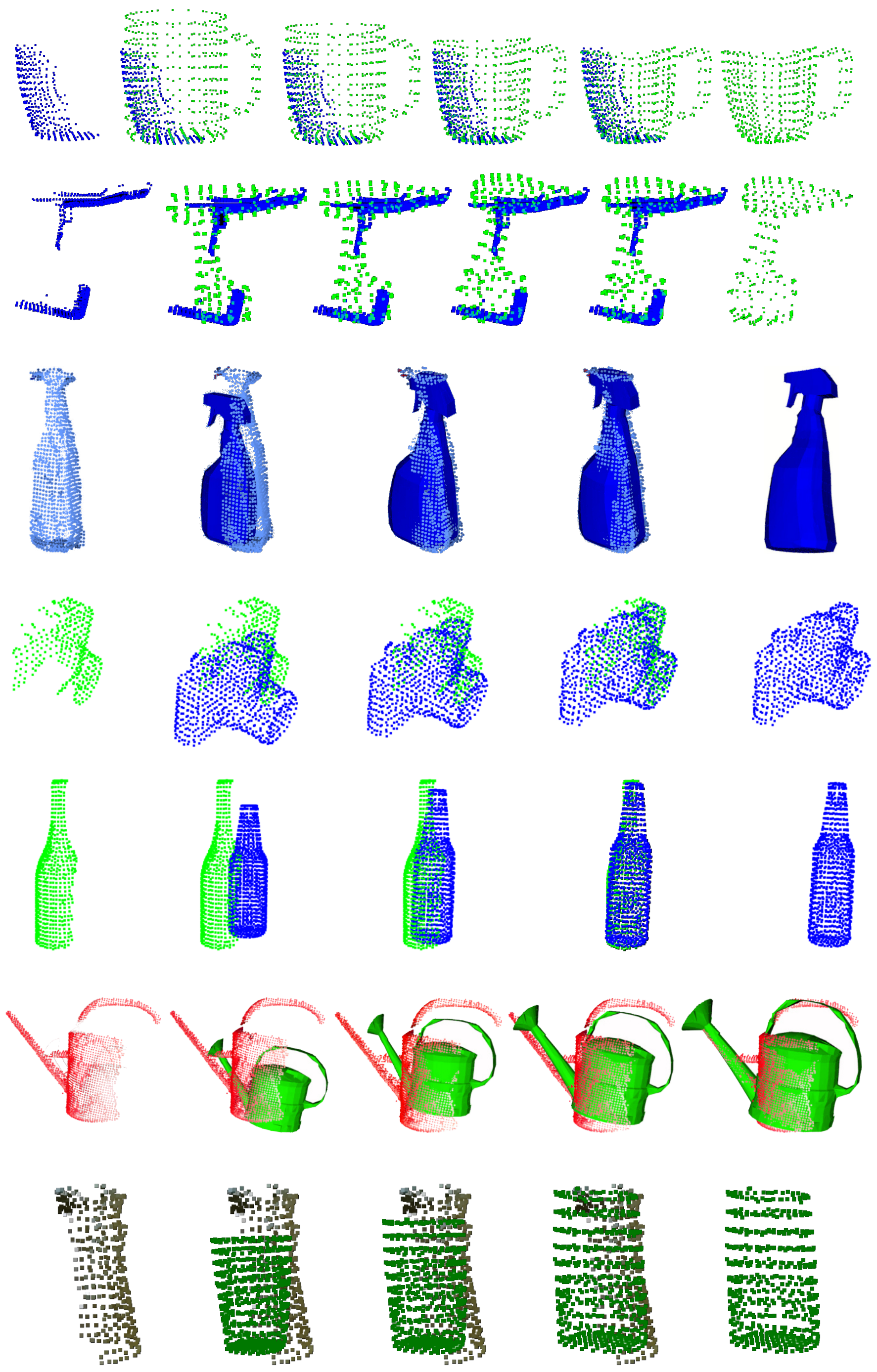

Figure 2.4: Shape registration. Canonical models are deformed to match the observed object instances. In each row, the input of the registration is shown at the leftmost column, while the results are presented on the right. Note the scale difference and the robustness of the method against misalignment (particularly evidenced by the camera, bottle and watering can registrations).

The Category-Level Shape registration method (CLS) [Rodriguez et al., 2018a] described here is compared against CPD [Myronenko and Song, 2010] for fully and partially observed shapes. It is expected, that for full views, CPD yields better results because CPD is employed to construct the shape space. To obtain the partial views, we used ray-casting on a single view of a tessellated sphere. CPD was parameterized with $\lambda = 2.0$ and $\beta = 2.0$ across all object instances. The same parameters were used for our method to construct the shape space. This ensures a fair comparison, because bad parameters for CPD will affect the quality of the shape spaces. All the shape spaces have $l = 5$ latent dimensions. The testing instances (T1 and T2 for each category) are completely novel, i.e., they are not used for building the shape spaces. The noiseless fully-observed shape is considered as the ground truth and the following registration error function is employed:

$$E(\mathbf{T}, \mathbf{C}) = \frac{1}{M} \sum_{m=0}^{M-1} \min_{n} \|\mathbf{T}_n - \mathbf{C}_m\|^2. \tag{2.13}$$

The registration error is computed as the averaged error across all partial views. The results are presented in Table 2.1.

The robustness of the category-level shape registration was evaluated against noise and misalignment, and the results were compared with the results given by CPD. Both noise and misalignment are evaluated gradually by increasing the level of noise and misalignment. Noise was added to each point of the observed instances by randomly sampling a point from a Gaussian distribution and scaling it by one of these noise factors: $[0.04, 0.08, 0.12, 0.16, 0.2]$. Each noise factor is used in a different noise level. Misalignment was achieved by including an additional rigid transformation to the pose of the

Table 2.1: Registration error of instances T1 and T2 for each object category. Comparison with CPD [Myronenko and Song, 2010]. Mean and (standard deviation) error values expressed in μm.

Instance	CLS (ours)	CPD
Camera T1	51.93 (10.45)	407.04 (491.89)
Camera T2	19.87 (4.59)	167.18 (358.16)
Bottle T1	25.92 (5.18)	140.51 (312.13)
Bottle T2	72.33 (11.35)	357.98 (536.81)
Spray Bottle T1	30.78 (1.89)	298.53 (409.14)
Spray Bottle T2	121.19 (19.16)	376.09 (720.73)
Drill T1	28.86 (1.42)	232.88 (1319)
Drill T2	58.50 (21.51)	216.35 (566.18)

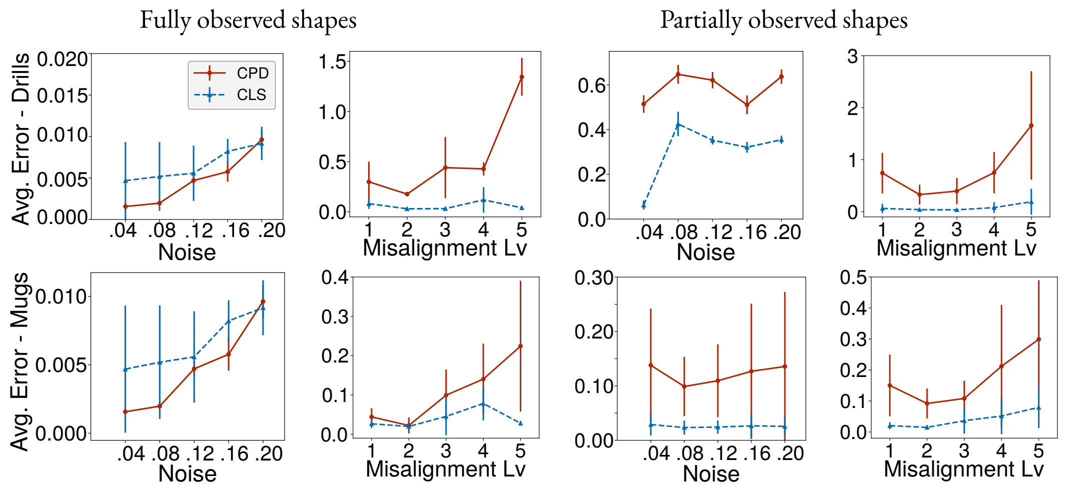

Figure 2.5: Shape registration evaluation against noise and misalignment. The category-level shape registration (CLS, dotted blue line) is compared with CPD (solid red line). The results of the *Drill* and *Mug* categories are presented in the first and second rows, respectively. The first two plots from left to right corresponds to the fully observed instances, whereas the last two plots corresponds to the partially observed instances. The CLS method described here outperforms CPD on partial views and misaligned fully observed shapes.

observed shape. For the translation, a three-dimensional unit vector was uniformly sampled and multiplied it by the factors: $[0.01, 0.02, 0.03, 0.04, 0.05]$, for each misalignment level, respectively. For the orientation, a three-dimensional unit rotation axis was uniformly sampled and joined with the angles: $[\pi/4, \pi/8, 3\pi/16, \pi/4, 3\pi/8]$. The rigid transformation was performed employing the axis-angle representation.

For each noise and misalignment level, the full view and six different partial views are considered. The registration errors are plotted in Fig. 2.5. For fully observed instances, CPD yields better lower registration errors when adding noise. As stated above, this is expected because CPD is the backbone for building the shape spaces. However, when the observed instance is misaligned, the category-level shape registration outperforms CPD. This is explained by the additional rigid transformation incorporated in Eq. (2.11). In addition, when the object is partially occluded or observed, the CLS method outperforms CPD due to the category-level information that lies in the shape space, which is not available in CPD.

2.6 Discussion

In this chapter, the shape space was introduced as a low-dimensional latent space spanned by the principal components of the deformation fields observed in the training instances of an object category. To build the shape space, a canonical model is chosen which represents a standard sample of the category. The non-rigid registration is performed by searching in the shape space for a shape descriptor whose deformation field applied to the canonical model matches at best with the observed instance. This search is formulated as a non-linear optimization that considers objects deformations, through the shape space, and minor misalignments, by incorporating a rigid transformation at global level.

In the evaluation, the registration method presented in this chapter demonstrated its capabilities to infer occluded or unseen parts of the objects thanks to the information encoded in the shape space. In addition, the robustness of the method against misalignments was shown which highlighted the contribution of the rigid transformation in the objective function of the optimization.

This non-rigid registration has been used in different object categories in multiple scientific projects applied for transferring grasping skills between a known canonical model and novel instances. The range of object categories demonstrates the expressiveness and applicability of the presented method. The application to skill transfer is presented in Chapter 4. In addition, the shape space has been also used to infer deformation vectors of unseen parts when the instances are observed only by RGB images, removing the dependency on depth data and the corresponding disadvantages of depth sensors, e.g., lower resolution, lower frame rate, poor performance for shiny and transparent objects, among others. This is presented in Chapter 3.

A software release that includes the training and inference of the shape registration method is available online [5]. We hope this code release will help the scientific community that is working in similar research directions. A video showing some capabilities of the released software is also available online [6].

[5] `https://github.com/AIS-Bonn/shape_registration.git`

[6] Video of code release: `https://zenodo.org/record/3991986/files/shape_space_code_release.mp4`

3 Category-Level 3D Non-Rigid Registration from RGB Images

"Intelligence is the ability to adapt to change."
— Stephen Hawking

In Chapter 2, we model typical object geometrical variations by shape (latent) spaces based on the observation that objects belonging to a category share similar extrinsic geometries. Consequently, a category-level non-rigid registration approach was developed that is able to deform a canonical model into an observed instance. The registration was based on 3D point sets which are captured in real robotic systems by 3D sensors such as laser scanners and RGB-D cameras.

The use of 3D sensors, however, presents several difficulties, especially for robotics applications such as grasping. Objects with transparent and shiny surfaces are, in general, challenging to perceive with 3D sensors. To illustrate this issue, two spray bottles with transparent surfaces were captured with the Azure Kinect sensor [1], one of the latest RGB-D cameras available, which is an improved version of the widely used Kinect V2 [Fankhauser et al., 2015] sensor for robotics applications. The observed spray models are depicted in Fig. 3.1. Note the missing points of both spray bottles and the faulty measurements produced by the liquid inside. The non-rigid registration of these instances is very challenging not only because of the missing points, but also due to the faulty measurements, since such data does not represent the real geometries.

In addition, 3D sensors impose constraints on the object proximity to the sensor (typically larger than 0.5 m) and on the lighting conditions (mostly indoors). Compared to RGB cameras, depth sensors have limitations regarding image resolution, field of view, and frame rate, which makes hard to perceive small, thin and fast moving objects. In order to address these issues, this chapter presents an approach to perform category-level non-rigid registrations from single-view RGB images.

The non-rigid registration from a single-view RGB image is a challenging problem. First, a mapping between 2D color pixels and 3D deformation vectors needs to be established

[1] https://docs.microsoft.com/en-us/azure/Kinect-dk/hardware-specification

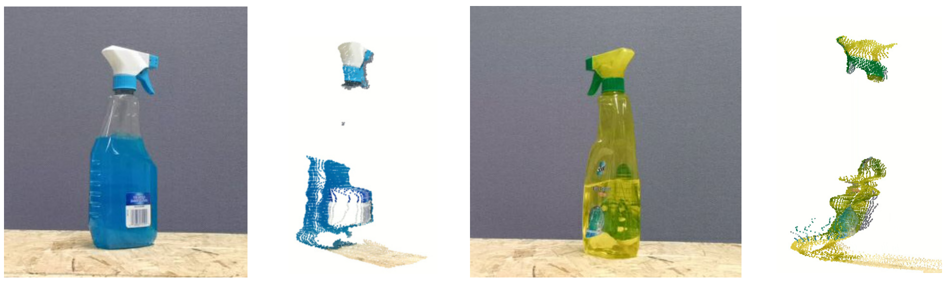
Figure 3.1: Issues perceiving transparent surfaces by RGB-D cameras (the data presented was captured by the Azure sensor was used). Several points are missing and faulty measurements are generated by the liquid inside.

without depth information. This is particularly difficult because the given input is not informative enough to infer the depth of the objects. For instance, in the case of estimating the height of the blue spray bottle of Fig. 3.1, i.e., the distance from the table to the top white surface, when the object is only observed from a top perspective. Second, from a single-view RGB image, an instance is not fully observable, therefore, the full shape needs to be reconstructed. However, this reconstruction problem is ambiguous, i.e., there exist several plausible shapes that explain the observed image.

The non-rigid registration from RGB images described in this chapter is possible thanks to prior knowledge modeled as shape spaces and Convolutional Neural Networks (CNNs). To train the shape spaces and the CNNs, textured realistic 3D object models are employed. The shape space of a category is built as explained in Sec. 2.3. The main contribution of the shape space is to infer deformation fields of the unseen parts. As result, the observed instance is reconstructed. The CNNs are trained by synthetic data. Ground truth 3D deformations are computed using the Coherent Point Drift (CPD) [Myronenko and Song, 2010] and represented as 3-channel feature maps (Sec. 3.2.1). In this manner, given a single-view RGB image, a CNN infers deformation vectors of the visible points, and these initial deformations are used by a shape space to infer a deformation vector for each point of the canonical model, including the occluded parts. The final result of the non-rigid registration is a dense deformation field. This allows to deform control points even after training, and thus, the method proposed here can be used for skill transfer, as presented in Chapter 4.

Figure 3.2 shows the canonical model and an instance of the spray bottle category. The testing object is observed by a RGB image. As result of the registration, the instance is reconstructed. The morphed canonical model is presented in a different pose to exhibit the quality of the reconstruction.

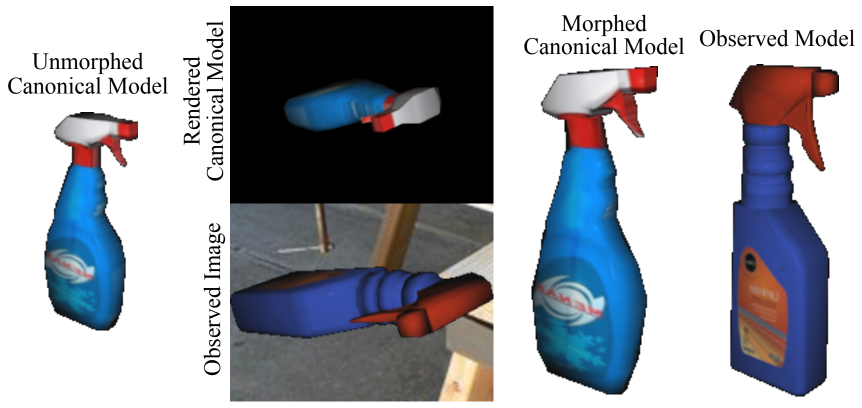

Figure 3.2: A canonical model of an object category is deformed based on a single-view RGB image. The model is reconstructed without any depth information and partially observing the object. Models are shown at the same scale to show the large deformation that the canonical model has undergone.

This chapter gives the argumentation published in [Rodriguez et al., 2020]. An overview of the approach is available online in Video 3.1 [2]. Animations of the results of non-rigid registrations performed with real data are also shown in the video.

3.1 Related Work

Rendering for Deep Learning

Deep learning approaches require a vast amount of data for training. In order to train models that can be used in real robotic applications, large datasets need to be created. However, the collection and annotation of input and target data is very time consuming. The use of synthetic data is a promising alternative to generate training samples in a time effective manner. There have been several successful examples that demonstrate the transfer of models trained with synthetic data into real world applications, including object detection [Tremblay et al., 2018a], semantic segmentation [Schwarz and Behnke, 2020; Zhang et al., 2017] and pose estimation [Xiang et al., 2018; Zhang et al., 2017]. One initial attempt to use rendered images for training deep neural networks was achieved by Tobin et al. [2017], who were able to transfer a neural network for object detection into real world applications using only synthetic data. For 6D pose estimation, recently Tremblay et al. [2018b] achieved state-of-the-art standards by using only rendered images. Similarly, pose refinement approaches such as [Li et al., 2018] and [Periyasamy et al., 2019], have shown that learning a mapping between 2D pixels and pose corrections is an effective strategy for the rigid registration problem. The approach presented in this chapter was

[2] Video 3.1: `https://zenodo.org/record/3991986/files/chapter_3.mp4`

partially inspired by these pose refinement approaches. Apart from [Rodriguez et al., 2020], that is mainly described in this chapter, to the best of the author's knowledge, there has not been any works that addressed the 3D non-rigid registration problem using deep learning trained on synthetic data.

Non-Rigid Registration

As described in Sec. 2.1, several methods that address the non-rigid registration problem of 3D objects assume the models are fully observable. In order to deal with incomplete or occluded data, low-dimensional latent spaces have been introduced [Burghard et al., 2013; Rodriguez et al., 2018a], as the method described in Chapter 2. Due to the knowledge represented in the latent spaces, non-observable object parts are reconstructed and plausible shapes are generated. However, such approaches require depth information, which is difficult to obtain for certain object surfaces. Thus, the approach presented in this chapter addresses the non-rigid registration problem based on RGB images, which do not present the difficulties associated with RGB-D data.

Based on RGB images, most of the work on non-rigid registration is directed to medical imaging applications. Krebs et al. [2017] have trained a Reinforcement Learning (RL) agent capable of performing image registration with state-of-the-art results. Li and Fan [2017] trains a fully convolutional neural network in a self-supervised manner to match two input brain images based on a similarity metric. From the robotics community, Huang et al. [2015] uses RGB data to enrich the registration based on regions of interest established by the appearance. However, 3D input data is still required. None of these works, nonetheless, are able to register 3D object models based only on RGB images.

Shape Reconstruction

The 3D shape completion problem have been tackled traditionally by radial basis functions [Carr et al., 2001], surface primitives [Kazhdan and Hoppe, 2013] and optimization problems [Nealen et al., 2006]. Recently, novel data-driven approaches have been proposed to solve this task. Choy et al. [2016] makes use of a large 3D object dataset in order to learn a map from 2D images to 3D shapes. The mapping is represented as a Recurrent Neural Network (RNN). Dai et al. [2017] are able to reconstruct object shapes by using an intermediate low-resolution output shape inferred by a CNN and a 3D-encoder predictor that outputs the final reconstructed object shape. Using a single depth image, Rock et al. [2015] proposed a method to complete 3D models by exploiting symmetries and by incorporating an exemplary database for training. From a single RGB image, in Wu et al. [2018], 3D shapes are reconstructed by combining deep generative models with adversarially learned shape priors.

Similar to the approach presented in this chapter, the methods introduced by Wu et al. [2018] and Choy et al. [2016] also reconstruct 3D models given RGB images. However,

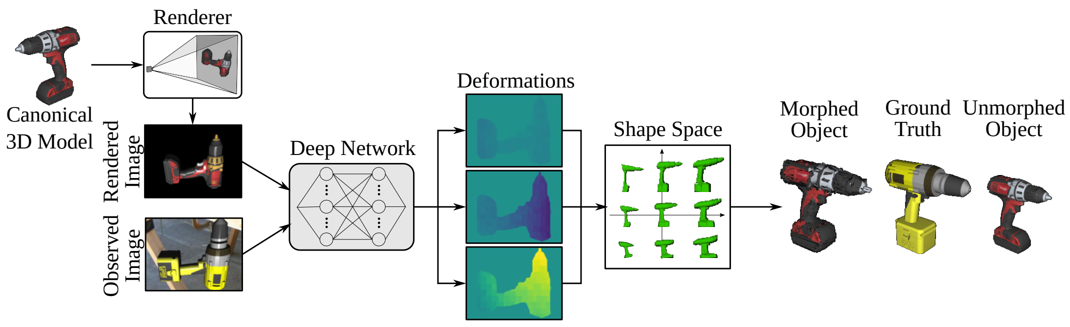

Figure 3.3: Overview of the non-rigid registration from RGB images. The 3D canonical model is rendered in the same pose as the observed instance. A deep CNN infers deformation vectors with the x, y, and z components per pixel of the rendered image. The deformations of the unseen parts are inferred by searching in the shape space for an instance that best matches the deformation given by the CNN. The object is reconstructed (morphed object) due to the knowledge encoded in the shape space. The ground truth of the observed object and the unmorphed canonical model are shown on the right.

their main goal is to generate plausible shapes, and they do not provide any kind of transformation or deformation field. Thus, those approaches can not be directly applied to skill transfer.

3.2 Non-Rigid Registration from RGB images

In this section, the category-level non-rigid registration from a single-view RGB image is introduced. The registration is performed between objects of the same category. In this manner, intra-class knowledge, e.g., typical geometrical variations, can be exploited to infer information of unseen or occluded parts. This knowledge is represented by means of Convolutional Neural Networks and shape spaces. The formulation of the approach presented here was motivated to be used for skill transfer between instances. Thus, a canonical model of an object category is deformed towards novel instances. Once the deformation between models is established, associated knowledge can be deformed in the similar manner, e.g., control poses for grasping.

The general approach is illustrated in Fig. 3.3. A CNN is trained to infer deformation fields given two input images, one from the canonical model and one from the observed instance. To provide the image of the canonical model, a renderer is employed. The initial deformation field given by the CNN serves as a target to match for a shape space search that ultimately defines the final deformation field and the corresponding result of the non-rigid registration.

In the following subsections, a formal deformation representation suitable for training the CNN is introduced. Later, we describe a zoom operation used to define a region of interest of the image and to increase the amount of details. Next, we argued on the network structure. And, finally, the inference of the final deformation field by the shape space is explained.

3.2.1 Deformation Representation

In contrast with the approach presented in Chapter 2, where models were represented as point sets discarding information such as the point color, in the approach presented here, the textures of the 3D models play a key role because the observed instances are mainly defined by their appearance in the RGB images. For this reason, object models are defined here as a textured three-dimensional meshes. However, the density of the mesh vertices is in general non-uniform. To address this issue, we associate a underlying point cloud generated by ray-casting against the mesh from several viewpoints on a tessellated sphere and by down-sampling by a voxel grid filter.

The mesh of the canonical model consists of M_m vertices. The point cloud matrix of the canonical model is denoted as $\mathbf{C} \in \mathbb{R}^{M\times 3}$, while the mesh vertices matrix as $\mathbf{C}_m \in \mathbb{R}^{M_m\times 3}$. The deformation between object instances are described by the Coherent Point Drift equations introduced in Sec. 2.2.1. The matrix of the deformed mesh $\mathbf{C}'_m \in \mathbb{R}^{M_m\times 3}$ is defined as:

$$\mathbf{C}'_m = \mathbf{C}_m + \mathbf{G}(\mathbf{C}_m, \mathbf{C}) * \mathbf{W}(\mathbf{C}, \mathbf{T}_i), \tag{3.1}$$

where $\mathbf{W}(\mathbf{C}, \mathbf{T}_i) \in \mathbb{R}^{M\times 3}$ describes the deformations that should be applied to the points of the canonical point cloud $\mathbf{C}$ to deform it to match the point cloud of the observed instance $\mathbf{T}_i \in \mathbb{R}^{N\times 3}$. The deformation matrix $\mathbf{W}(\mathbf{C}, \mathbf{T}_i)$ is multiplied by $\mathbf{G}(\mathbf{C}_m, \mathbf{C}) \in \mathbb{R}^{M_m\times M}$ to get the deformation of the mesh vertices. $\mathbf{G}(\mathbf{C}_m, \mathbf{C})$ is computed following Eq. (2.5) and establishes coherent motion of the vertices. Similar to the shape registration based on point sets, the uniqueness of the deformation between an observed instance and the canonical model is captured entirely by $\mathbf{W}(\mathbf{C}, \mathbf{T}_i)$.

For generating the target images for training, the matrix $\boldsymbol{\delta} \in \mathbb{R}^{M\times 3}$ is introduced:

$$\boldsymbol{\delta}(\mathbf{C}, \mathbf{T}_i) = \mathbf{G}(\mathbf{C}, \mathbf{C}) * \mathbf{W}(\mathbf{C}, \mathbf{T}_i). \tag{3.2}$$

The matrix $\boldsymbol{\delta}$ gives directly a deformation vector for each point of $\mathbf{C}$ in contrast to $\mathbf{W}$ that defines kernel weights, which can be interpreted as unconstrained deformation vectors. For training the CNN, a relation between pixel differences and object deformations is assumed. For this reason, target images are generated from $\boldsymbol{\delta}$ which represents object deformations explicitly. In other words, we avoid that the CNN learns $\mathbf{G}$. Furthermore,

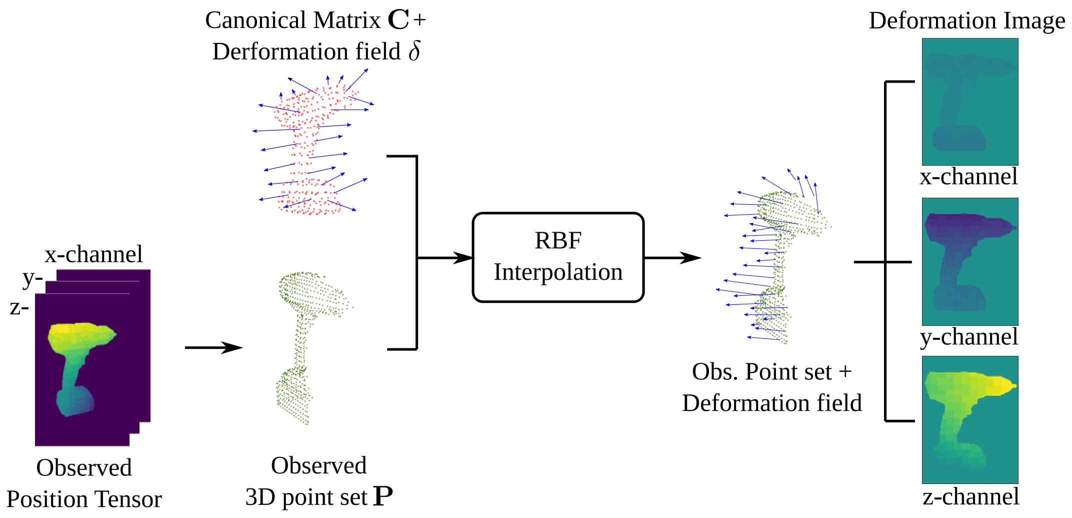

Figure 3.4: Generation of network target images. The renderer provides a three-channel feature map containing the position of each observed pixel. This image can be represented as a 3D point set $\mathbf{P}$. The canonical model $\mathbf{C}$ and the deformation field $\boldsymbol{\delta}$ initialize a RBF interpolator to find the deformation values for each point of $\mathbf{P}$. Finally, the interpolated deformation values are expressed as a three-channel feature map.

changes between values of $\boldsymbol{\delta}$ are smoother compared to changes in the values of $\mathbf{W}$. The smoothness is introduced by the regularization of the point movements enforced by $\mathbf{G}$.

The network targets are three-channel feature maps, where each channel contains the deformations in each of the coordinate axis: x, y and z. The mapping between the 3D deformation vectors and the deformation target images is facilitated by the renderer. Besides the color image, the renderer gives a three channel position map. Each pixel of this map provides a position value as result of a baycentric interpolation between the position of the closest mesh vertices of the visible parts of the rendered model. These position values can be represented as a 3D point set $\mathbf{P}$. Given the matrix of the canonical model $\mathbf{C}$ and the associated deformation field $\boldsymbol{\delta}$, that provides a deformation vector for each point of $\mathbf{C}$, a Radial Basis Function (RBF) interpolation with a linear kernel is employed to calculate the deformation of the points on $\mathbf{P}$. A separate interpolator is used for each coordinate axis. An overview of the network target image generation is shown in Fig. 3.4.

3.2.2 Learning Deformation Fields by CNNs

The proposed CNN aims to infer deformation values for the visible pixels. The network receives as input two images: one containing the observed model and the second with a rendered canonical model. Both images are aligned according to the object pose. The alignment encourages the network to concentrate only on the pure deformation of the objects rather than large differences in their poses.

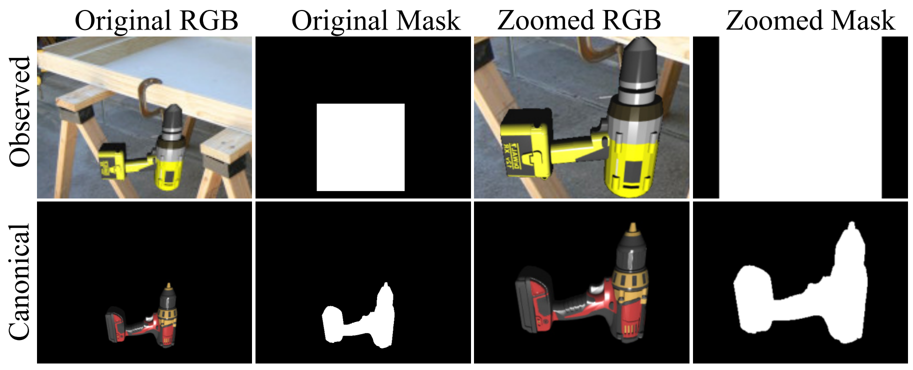

Figure 3.5: Image preprocessing for network input. The renderer creates an image of the canonical model at the same pose as the observed instance. Both images are cropped according to the maximum bounding box that encapsulates both objects and preserves the aspect ratio of the network input. The images are finally cropped and bilinear upsampled.

In order to increase the amount of informative pixels passed to network, i.e., the amount of object details, the images of the observed and canonical instances are zoomed in and cropped to match the input size of the network (256 × 192 pixels in the experiments presented here). This allows the network to concentrate in the interesting parts of the observed image neglecting uninteresting background pixels. To generate the rendered image, the canonical mesh is placed at the same pose as the observed instance w.r.t the virtual camera. Then, the maximum bounding box that encapsulates both objects and preserves the image size ratio of the network input images is found. Finally, both images are cropped according to this bounding box and bilinearly upsampled to the input size of the network. The knowledge of the object category and pose can be given by off-the-shelf semantic segmentation [Schwarz et al., 2018], object detection [Tremblay et al., 2018a] and object pose estimation [Periyasamy et al., 2019; Xiang et al., 2018] methods. The zoom operation of an object instance is shown in Fig. 3.5.

Based on the assumption that the optical flow between two images is related with the object deformation (Eq. (3.1)) to be applied to match two objects, the FlowNet2 [Ilg et al., 2017] network architecture is proposed as the backbone of the CNN presented here. FlowNet2 was adapted to receive as input two RGB images with their respective masks. The mask of the canonical image is given by the renderer and tells the network which pixels are relevant in the image. The mask can be interpreted as a regularizer distinguishing the fore- and the background. The mask of the observed instance is defined as bounding box encapsulating the object. The masks can be considered as an additional channel of the images. Therefore, in the remaining sections of this chapter, the upcoming references to observed and canonical images refer to four-channel images. Finally, the FlowNet2 output is also modified such that the network returns a three-channel feature map representing the deformation values.

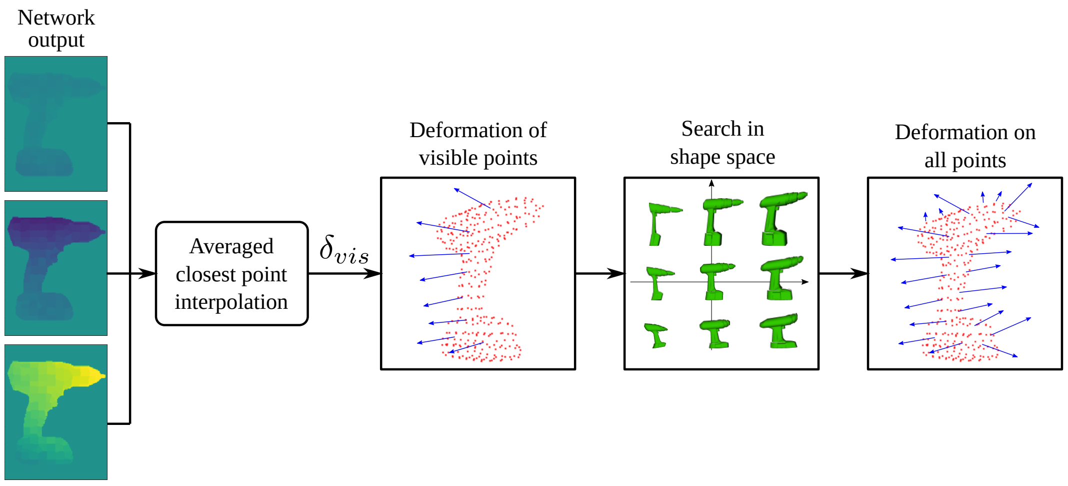
Network output
Averaged closest point interpolation
δ_{vis}
Deformation of visible points
Search in shape space
Deformation on all points

Figure 3.6: Deformation field inference based on deformation images. Given the inferred deformation images by the network, an averaged closest point interpolation assigns deformation vectors to each of the closest points of the visible points to $\mathbf{C}$. This can be interpreted as finding a deformation for the visible parts. The deformation of all points on $\mathbf{C}$ are inferred by searching in the shape space for a shape descriptor that matches at best with the deformation of the visible points.

3.2.3 Deformation Inference of Non-Observable Parts

In single-view RGB images, objects can not be fully observed. However, in order to register two objects non-rigidly, deformation vectors need to be provided for all point (including the non-observable point) of the deforming object. By means of shape spaces (Section 2.3), deformation fields of the non-observable parts are inferred. The overall problem is addressed as follows: from a RGB image, deformation fields of the visible parts are inferred by a CNN as explained in Section 3.2.2; the inferred deformation field serves as a target of an optimization problem that searches in the shape space of the object category for a plausible deformation field that matches the inferred deformation field at best. This optimization problem is inspired by the inference of deformation fields from partial views given 3D data mentioned in Section 2.4 and published in [Rodriguez and Behnke, 2018; Rodriguez et al., 2018a]. The deformation field for all points (including non-observable parts) on $\mathbf{C}$ is depicted in Fig. 3.6

Each of the pixels of the three-channel deformation images given by the CNN can be represented as a three-dimensional deformation vector of the corresponding 3D point of the position tensor given by the renderer. In other words, the position tensor can be represented as a 3D point set, in which each 3D point has an associated 3D deformation vector given by the CNN. Note that the number of points of this deformation field is different than the number of points of $\mathbf{C} \in \mathbb{R}^{M\times 3}$ and therefore it can not be directly transformed to the shape space whose mapping matrix $\mathbf{L}$ (Eq. 2.10) expects the same number of points as the canonical point cloud $\mathbf{C}$. Thus, the deformation field given the CNN needs to be mapped to deformation vectors to the points of the canonical model. For this, the closest point to $\mathbf{C}$ for each 3D position vector (position tensor pixel) is found. In general, the number of visible points in the position tensor is higher than the number of visible points of $\mathbf{C}$, i.e., a point on $\mathbf{C}$ is assigned to multiple points of the position tensor. The mean deformation value of all assigned points for each visible point on $\mathbf{C}$ defines ultimately its deformation vector. The resulting matrix $\boldsymbol{\delta}_{vis} \in \mathbb{R}^{M\times 3}$ describes how the closest points of the visible points to $\mathbf{C}$ should be moved to matched the observation given by the RGB image. The deformations of points without any assigned closest point are set to zero and correspond with the non-observable points. The process of finding δ_{vis} is referred here as averaged closest point interpolation.

A search in the shape space for a feature descriptor $\mathbf{x}$ whose deformation field is the closest (in a least-square sense) to the field of the visible points will define the final deformation field that is the result of the non-rigid registration. The deformation field represented by the feature vector $\mathbf{x}$ can be expressed as:

$$\hat{\boldsymbol{\delta}} = \tilde{\mathbf{G}}(\mathbf{C}, \mathbf{C}) * (\mathbf{L} * \mathbf{x} + \bar{\mathbf{w}}), \tag{3.3}$$

where $\tilde{\mathbf{G}}(\mathbf{C}, \mathbf{C}) \in \mathbb{R}^{3M\times 3M}$ is a rearranged form of $\mathbf{G}(\mathbf{C}, \mathbf{C}) \in \mathbb{R}^{M\times M}$ with additional zeros to match the dimensionality of $\mathbf{L} * \mathbf{x} + \bar{\mathbf{w}} \in \mathbb{R}^{3M\times 1}$. The feature vector $\mathbf{x}$ is found by optimizing the following loss function between $\boldsymbol{\delta}_{vis}$ and $\hat{\boldsymbol{\delta}}$:

$$L(\boldsymbol{\delta}_{vis}, \hat{\boldsymbol{\delta}}) = \sum_{v \in i_{vis}} \left\| \boldsymbol{\delta}_{vis}(v,.) - \hat{\boldsymbol{\delta}}(v,.) \right\|^2, \tag{3.4}$$

where i_{vis} is the set of indices of the visible points and $\boldsymbol{\delta}(v,.)$ represents the v-th row of the matrix $\boldsymbol{\delta}$. For computational efficiency, the optimization is reformulated as a least-squares problem. Let M_v denote the number of visible points and $\overline{\boldsymbol{\delta}}_{vis} \in \mathbb{R}^{1\times 3M_v}$ the row vector resulting by removing all non-observable points of $\boldsymbol{\delta}_{vis}$ followed by the respective arrangement. Similarly, the matrix $\mathbf{D} \in \mathbb{R}^{M_v \times M}$ is introduced to delete the same points rows from $\hat{\boldsymbol{\delta}}$ as the ones removed from $\boldsymbol{\delta}_{vis}$. Thus, minimizing Eq. (3.4) can now be expressed as:

$$\begin{aligned} L(\overline{\boldsymbol{\delta}}_{vis}, \mathbf{x}) &= \left\| \overline{\boldsymbol{\delta}}_{vis} - \mathbf{D}\boldsymbol{\delta} \right\| \\ L(\overline{\boldsymbol{\delta}}_{vis}, \mathbf{x}) &= \left\| \overline{\boldsymbol{\delta}}_{vis} - \mathbf{D}(\tilde{\mathbf{G}}(\mathbf{C}, \mathbf{C}) * (\mathbf{L} * \mathbf{x} + \bar{\mathbf{w}})) \right\|^2. \end{aligned} \tag{3.5}$$

Let define:

$$\mathbf{A} := (\mathbf{D}\tilde{\mathbf{G}}(\mathbf{C}, \mathbf{C}) * \mathbf{L}), \tag{3.6}$$

and

$$\mathbf{B} := (\overline{\boldsymbol{\delta}}_{vis} - \mathbf{D}\tilde{\mathbf{G}}(\mathbf{C}, \mathbf{C}) * \bar{\mathbf{w}}), \tag{3.7}$$

such that the optimized feature vector $\mathbf{x}^*$ that is the lower representation of the deformation field $\hat{\mathbf{W}}^*$ for all points in $\mathbf{C}$ is expressed by:

$$\mathbf{x}^* = \arg\min_{\mathbf{x}} L(\overline{\boldsymbol{\delta}}_{vis}, \mathbf{x}) = \arg\min_{\mathbf{x}} \|\mathbf{A}\mathbf{x} - \mathbf{B}\|^2. \tag{3.8}$$

By replacing $\hat{\mathbf{W}}^*$ in Eq. (3.1), the shape of the observed model is reconstructed.

3.3 Dataset

As explained above, this chapter presents a data-driven non-rigid registration approach. Specifically, a CNN and a shape space are trained to infer deformation fields. The training process requires 3D object models of the categories. Additionally, annotations about the deformations between objects are required for training the CNN. Although available 3D object datasets such as ShapeNet [Chang et al., 2015] and MeshNet [Wu et al., 2015] provide a collection of 3D objects of different categories, no information about the deformation between objects is provided. Thus, in this section a new dataset is introduced that is suitable for training deformation fields in a supervised manner based on RGB images.

The dataset contains a collection of 3D textured models that are employed to generate a set of rendered color images and corresponding ground truth deformations. The texture of the object plays a key role in this approach because the models are only observed by RGB images, i.e., the objects are entirely represented by their appearance. Due to the difficulty and time considerations of building the dataset using real data, a renderer is used to generate the training images. For this reason, it is critical that the 3D models are in fact realistic object models, such that this approach can be implemented in real robotic applications. The models were obtained from an online database[3] and from [Wang et al., 2019], who recently released an aligned 3D dataset with objects relevant for robotics based on models from ShapeNet [Chang et al., 2015].

From the collected 3D models, one model from each category is selected as the canonical one. The remaining models compose the training instances. The target images, i.e., the ground truth deformations, are calculated using CPD (Sec. 2.2.1). The mapping between the deformation fields of the 3D point set to the image space is accomplished by radial basis function interpolation as explained in Sec. 3.2.1. The training images are generated by rendering the 3D models from different viewpoints. Samples of the training dataset are shown in Fig. 3.7. The shape space of the category is also trained using the collected set of 3D object models.

[3] https://sketchfab.com

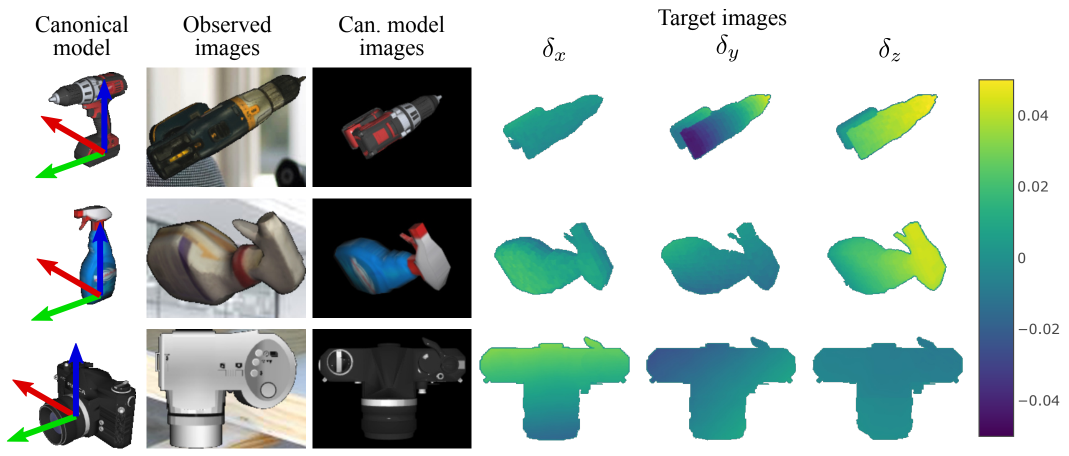

Figure 3.7: Samples of the training dataset for learning deformation fields from RGB images. The canonical models with their reference frames are shown at the leftmost column. The network input is composed of four images: a RGB image of the training instance with its bounding box mask, and an image of the canonical model with its mask. The target feature maps represent the x, y and z components of the deformation fields. The targets are visualized by heatmaps, where the background is painted white for clarity.

It is well established in the deep learning community, especially for data-driven approaches, as the one presented here, that the amount of data can limit the capabilities of the network. This is particularly problematic for the method presented here because the number of available realistic 3D models, especially graspable objects, is limited. To address this issue, the dataset is augmented by generating additional 3D models as the result of interpolating the deformation fields between an instance and the canonical model (Fig. 3.8).

The interpolated models are generated as follows. First, the canonical model is registered non-rigidly towards all training instances by using CPD (Eq.(2.8)). As result, $\mathcal{T}(\mathbf{C}, \mathbf{W}_i)$,

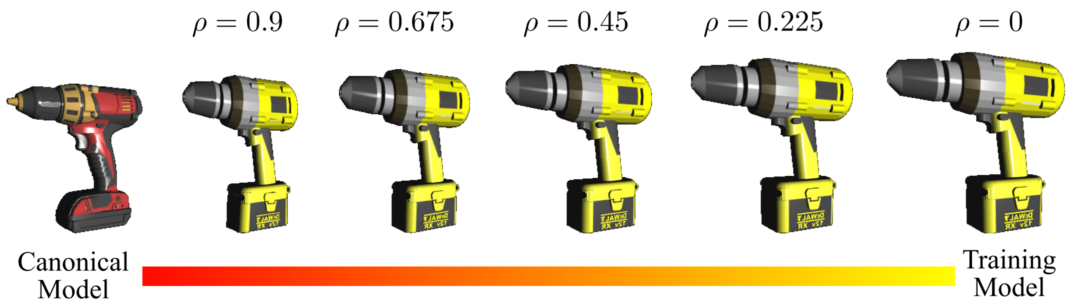

Figure 3.8: Instance generation of novel 3D objects by interpolation between existing models. The texture of the observed model is preserved to avoid keeping the texture of the canonical model in all rendered images.

and most importantly, the deformation field $\mathbf{W}_i$, are calculated for each instance. Novel models can now be generated by interpolating each $\mathbf{W}_i$. However, this results in models with the same texture as the canonical model, which limits the generalization capabilities of the network. For this reason, the interpolation needs to be performed from the training instance to the canonical model such that the texture of the training instance is transferred to the interpolated model. Thus, the following inverse transformation from a training instance $\mathbf{T}_i$ based on the deformation field $\mathbf{W}_i$ is formulated:

$$\mathcal{T}^{-1}(\mathbf{T}_i, \mathbf{W}_i) = \mathbf{T}_i + \mathbf{G}(\mathbf{T}_i, \mathbf{C}) * (-\mathbf{W}_i). \tag{3.9}$$

Eq. (3.9) allows to deform the training instances without performing CPD avoiding computationally expensive operations. The already calculated $\mathbf{W}_i$ defines a deformation field but it is now applied to the training instance instead of the canonical one by computing an additional Gaussian kernel matrix $\mathbf{G}(\mathbf{T}_i, \mathbf{C})$. This matrix can be interpreted as a mapping of deformation fields from $\mathbf{C}$ to $\mathbf{T}_i$. In a similar manner to Eq. (3.1), the mesh vertices $\mathbf{T}_{m,i}$ of the training instance are transformed by the expression:

$$\mathbf{T}'_{m,i} = \mathbf{T}_{m,i} + \mathbf{G}(\mathbf{T}_{m,i}, \mathbf{C}) * (-\mathbf{W}_i). \tag{3.10}$$

Finally, by adding an interpolation factor ρ, Eq. (3.10) is expanded as:

$$\mathbf{T}'_{m,i}(\rho) = \mathbf{T}_{m,i} + \mathbf{G}(\mathbf{T}_{m,i}, \mathbf{C}) * (-\rho * \mathbf{W}_i). \tag{3.11}$$

Following Eq. (3.11), all training instance are morphed given the interpolation parameter ρ. The input training images are generated by rendering the canonical model together with the interpolated models from different viewpoints. The masks for the canonical model and the bounding box are computed and all the images are aligned and zoomed in as described in Sec. 3.2.2. The ground truth deformation fields are generated by computing the following equation:

$$\boldsymbol{\delta}(\mathbf{C}, \mathbf{T}_i) = \mathbf{G}(\mathbf{C}, \mathbf{C}) * (1 - \rho) * \mathbf{W}_i, \tag{3.12}$$

which include the interpolation parameterρ. The target images are created by the Radial Basis Function interpolation presented in Sec. 3.2.1.

3.4 Evaluation

The approach presented in this chapter is tested on four different categories: *spray bottles*, *cameras*, *bottles*, and *drills*. The collection of 3D models are composed by 12, 15, 14, and 16 instances, respectively. Figure 2.3 presents some of the instances belonging to the categories. The canonical model for creating the images and the shape spaces is shown in the leftmost column. The dataset is augmented with three interpolated models, i.e.,

$\rho \in \{0.0, 0.25, 0.5, 0.75\}$, $\rho = 0$ means no interpolation. The 3D models are rendered from 74 viewpoints on a tessellated sphere. Each category is tested on 2 instances (T1 and T2). All together, accounting the number of 3D models (including the interpolated ones) and the number of viewpoints, results in 2664, 3552, 3256 and 3848 training images, respectively. Instances of training images are shown in Fig. 3.7. Consequently, the testing dataset is composed by 552 images for each category. The CPD method is employed to build a shape space for each category and to calculate the ground truth deformation images. The CPD method is parameterized with $\lambda = 2.0$ and $\beta = 2.0$ for all non-rigid registrations. The shape spaces are built with $l = 5$ latent dimensions. Neither the shape space nor the CNN are trained with the testing instances.

For training the CNN, the dataset is divided randomly, 90% of the images are taken for training while the remaining 10% validates the model. The target feature maps are scaled by 1000 to increase the difference between background and foreground pixels. The network aims to optimize the pixel-wise L_2 loss. A scheduler halves the learning rate every 600 epochs. The initial learning rate equals 3×10^{-5} and it is bounded to 1×10^{-6}. The training is performed on two Graphics Processing Units (GPUs) with batch size equal to 24 (effectively 48 in total). The training is stopped after 2000 epochs.

3.4.1 Experimental Results

The category-level non-rigid registration presented in this chapter is evaluated according to the following reconstruction error function:

$$E(\mathbf{T}, \mathbf{C}') = \frac{1}{m} \sum_{i=0}^{m-1} \min_{j} \|\mathbf{T}(i,.) - \mathbf{C}'(j,.)\|^2, \tag{3.13}$$

where $\mathbf{C}' \in \mathbb{R}^{n \times 3}$ is the deformed canonical model result of the registration and $\mathbf{T} \in \mathbb{R}^{m \times 3}$ is the ground truth observed model. The error function is defined as the average of the closest distance for each point of the ground truth to the deformed model represented as a point set.

Samples of inferred deformation images of the spray bottle and camera categories are shown in Fig. 3.9. The performance of the registration from RGB images is compared with other registration methods: the Category-Level Shape registration (CLS) [Rodriguez et al., 2018a] described in Chapter 2 and the CPD [Myronenko and Song, 2010]. In contrast with these methods, our method does not have access to 3D data. Both, the CLS and CPD methods are parameterized with the same values as our method, i.e., $\lambda = 2.0$, $\beta = 2.0$, and $l = 5$. This makes a fair evaluation because bad parameters for CPD will affect the quality of the generated training images and the performance of the shape space, and consequently the final performance of our method. The input of our method are rendered images from 74 different viewpoints. For CPD and CLS, the input point sets are

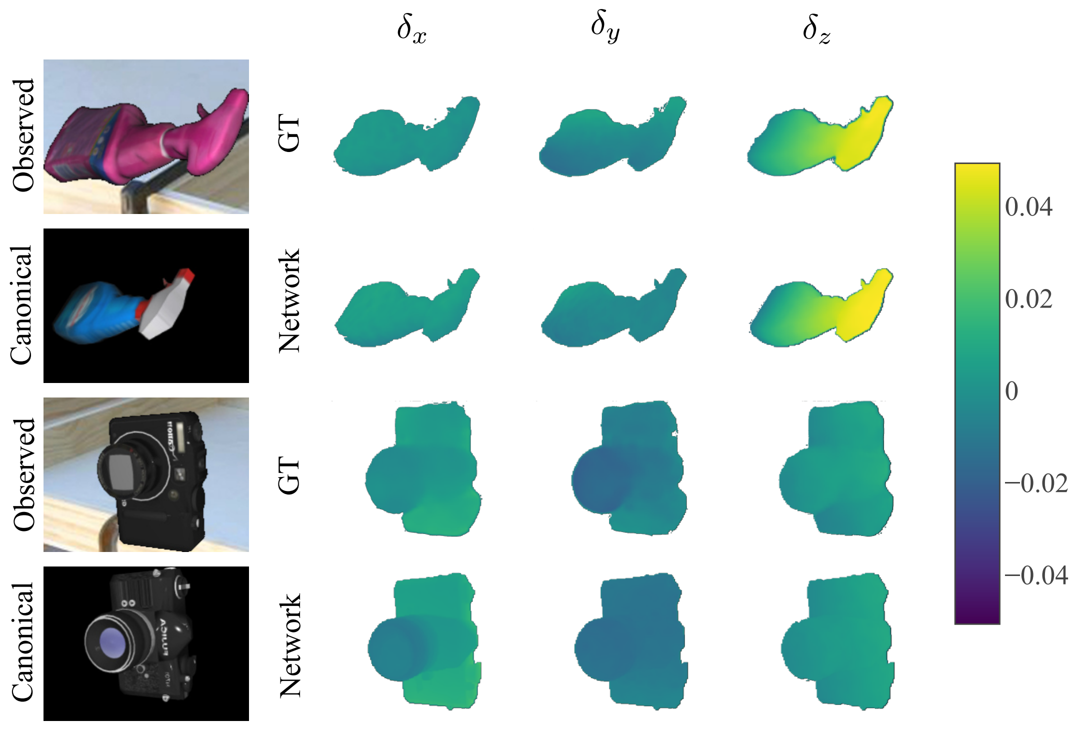

Figure 3.9: Inferred deformation fields (in meters) by the CNN. The deformations are visualized as heatmaps, where the background pixels have been colored white for clarity. The ground truth (GT) target images are also presented for comparison.

generated by ray-casting the 3D models from the same observed viewpoint as the rendered images.

In addition to the comparison with the CPD and CLS methods, the reconstruction error is also calculated to the deformed models resulting from the ground truth target images. This error can be interpreted as a bound for the reconstruction error of our CNN, and thus, it allows to evaluate the performance of the CNN. The non-rigid registration is performed for all images in the testing set. For each method, the mean and standard deviation of the reconstruction errors for each testing instance are presented in Table 3.1. The CLS and CPD methods were already compared in Section 2.5, but the results are shown here again for completeness. The lowest reconstruction errors are achieved by the CLS method. This is rather expected because it incorporates the same shape space as our approach, and CLS has access to depth data. The difference of the reconstruction error between CLS and our method can be attributed to the performance of the network. This can be observed by comparing the error of the CLS method with the ground truth.

Our non-rigid registration method outperforms CPD even without depth data. The lower reconstruction errors are achieved by the prior knowledge encoded in the shape space which is not present in CPD. In other words, our method is able to produce more

Table 3.1: Reconstruction error of the the non-rigid registration from RGB images compared to CLS [Rodriguez et al., 2018a] and CPD [Myronenko and Song, 2010]. Mean (and standard deviation) error values expressed in μm.

Instance	Ground Truth	CLS	CPD	Ours
Camera T1	34.61 (1.97)	51.93 (10.45)	407.04 (491.89)	102.17 (47.89)
Camera T2	16.45 (1.61)	19.87 (4.59)	167.18 (358.16)	18.80 (5.11)
Bottle T1	23.25 (2.34)	25.92 (5.18)	140.51 (312.13)	45.21 (9.75)
Bottle T2	90.42 (28.54)	72.33 (11.35)	357.98 (536.81)	88.35 (18.39)
Spray B. T1	29.84 (1.42)	30.78 (1.89)	298.53 (409.14)	47.87 (12.99)
Spray B. T2	111.94 (14.29)	121.19 (19.16)	376.09 (720.73)	154.97 (82.34.39)
Drill T1	21.18 (0.949)	28.86 (1.42)	232.88 (1319)	52.71 (23.54)
Drill T2	63.95 (5.23)	58.50 (21.51)	216.35 (566.18)	119.88 (107.43)

category-like shapes. The lower standard deviation indicates that the object viewpoint affect less our method than CPD.

The average inference time equals 7.42 s. This time, however, is dominated mainly by the closest point interpolation (5.34 s), and matrix multiplication for the inference of the occluded parts (1.96 s). Therefore, by optimizing the current implementation, this average time can be significantly reduced.

An additional evaluation with respect to misalignments is also carried out. A translation vector is added to the observed object pose for this purpose. This additional vector represents errors coming from pose estimators modules. The three components of the translation vector are sampled from a uniform distributions $\mathcal{U}(-0.05, 0.05)$. Testing

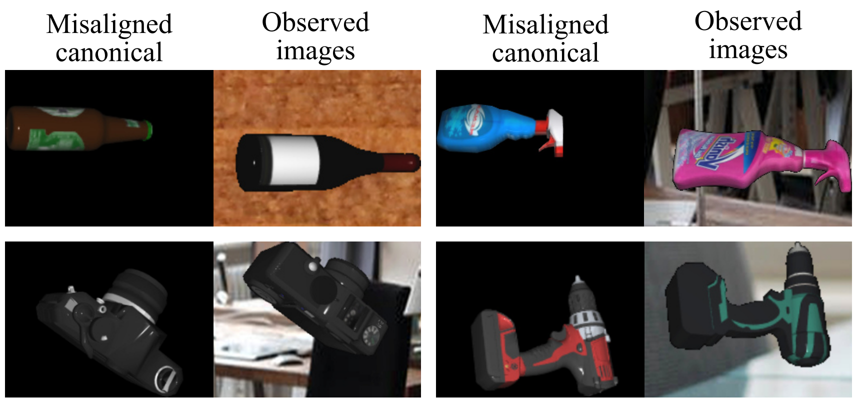

Figure 3.10: Testing images with additional misalignment.

Table 3.2: Reconstruction error with additional misalignment. Mean (and standard deviation) error values expressed in μm

Category	CPD	Ours
Camera T1	168.54 (357.8)	105.26 (64.21)
Camera T2	406.45 (492.03)	306.96 (127.89)
Bottle T1	297.79 (579.49)	227.90 (146.0)
Bottle T2	852.40 (1818)	289.36 (147.68)
Spray Bottle T1	1035 (406.69)	146.89 (117.57)
Spray Bottle T2	1488 (554.33)	255.69 (167.32)
Drill T1	232.35 (1325)	92.96 (58.23)
Drill T2	215.54 (565.48)	262.31 (228.40)

images with the additional translation vector are shown in Fig. 3.10. The reconstruction error of CPD and our method with the misalignments are presented in Table 3.2. Our method is affected by these large misalignments because pose refinement is not explicitly modeled. Nonetheless, our method is able to achieve lower reconstruction errors than CPD.

In order to demonstrate the applicability of the proposed approach, RGB images of real objects are captured and registered. Qualitative results are shown in Fig. 3.11. All the objects are presented by the first time to our approach. The 3D meshes of two spray bottles were available at the moment of the evaluation. These models are shown next to the reconstructed models as result of the non-rigid registration. For the pink spray bottle, the canonical model undergoes a large deformation (>10 cm in the z direction) considering its dimensions (21.3 cm height). On the other hand, the registration of the white spray bottle presents difficulties, presumably because spray bottles of this large size were never observed by the CNN. This suggests a further augmentation of the collection of 3D models by scaling existing meshes. A quantitative evaluation is also carried out. The two real spray bottles are observed from 12 different equal distributed poses by turning the objects by 30° from their standing position. The reconstruction errors are presented in Table 3.3. Once more, our method outperforms CPD, in this case, registering real objects.

Remarkably, our method is able to even register objects with transparent surfaces including drinking and spray bottles. These kind of objects are normally misperceived by 3D sensors as depicted in Fig. 3.1. This highlights the novelty and relevance of our method that can registers objects from single-view RGB images.

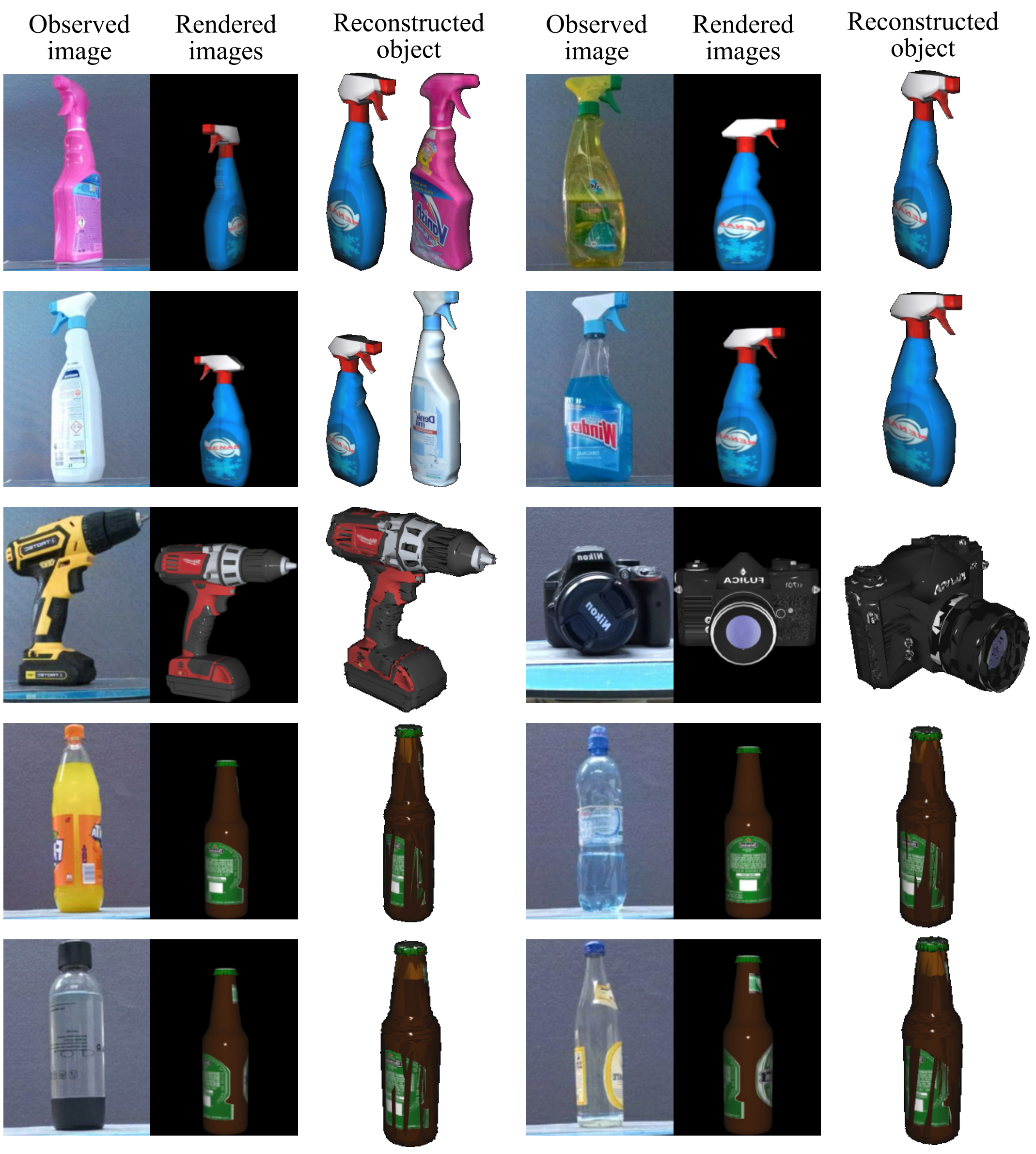

Figure 3.11: Results of the non-rigid registration on real RGB images. When available, the 3D ground truth object models are displayed next to the deformed instances. Our method is able to successfully register objects with transparent surfaces which are normally hard to perceive by 3D sensors.

Table 3.3: Reconstruction error of registration of real objects. Mean (and standard deviation) error values expressed in μm

Category	CPD	Ours
Real Spray Bottle - pink	451.78 (329.7)	67.34 (41.18)
Real Spray Bottle - white	436.38 (409.25)	398.03 (229.05)

3.5 Discussion

In this chapter, a novel approach to register objects from the same category in a non-rigid manner from single-view RGB images was presented. The use of RGB images as the only input is especially relevant with objects that are challenging to perceive such as those with transparent and shiny surfaces. It was demonstrated in the evaluation section how such objects can be registered by employing the method explained here. The method has mainly two phases: in the first one, a deformation field of the observable parts of the object is inferred by a CNN; and in the second one, a deformation field of the non-visible parts is inferred by searching in the shape space of the category for the deformation field that matches at best the inference given by the CNN. Because of the data-driven nature of the method, training images are generated by means of a renderer and a collection of realistic 3D models. As result of the registration, observed objects are reconstructed thanks to prior geometrical knowledge of the category represented in shape spaces.

The approach has been tested on different categories with synthetic and real data. In the evaluation, our method outperformed the Coherent Point Drift (that required depth data), even with misaligned objects.

As a natural step to enrich the method presented here, the object category can be incorporated in the CNN. In this manner, additional semantic segmentation approaches are not required. Furthermore, in order to improve the performance against noise on the observed pose, a pose refinement model should be integrated both in the CNN and in the shape space. Finally, color and texture registration between any two different instances can extend the applicability of this approach to scenarios where the visualization of objects is relevant, such as in teleoperation tasks.

4 Functional Grasp Skill Transfer

"The ideal condition
Would be, I admit, that men should be right by instinct;
But since we are all likely to go astray,
The reasonable thing is to learn from those who can teach."
— Sophocles

In previous Chapters 2 and 3, the non-rigid registration problem was addressed from RGB and 3D sensor input data. As result of the registration, the object models are reconstructed and deformation fields that warp a canonical model into the observed instances are inferred. The registration can be performed on partial views thanks to the prior knowledge embedded in the shape space. This chapter focuses on grasping skill transfer, i.e., given a known object with associated grasping information (e.g., pregrasp and grasp control poses), this grasping information is transferred to novel observed instances. The transfer is described by the geometrical variations between the given and the observed instance by means of the category-level registration methods of the previous chapters. The transfer is done even to parts that are not observed due to the shape reconstruction of the shape space.

The grasping transfer approach presented here is inspired by human nature. When people interact with novel objects, i.e., objects that are presented for the first time, people are still able to dexterously operate these objects immediately, e.g., when using cutlery in a restaurant visited for the first time or by making holes with a drill never seen before. This transfer happens effortless in humans because of the past experiences with similar objects. The manner how the objects are grasped and manipulated is adapted according to the object geometry.

The approach presented in this chapter transfers *functional* grasping skills. As result, a grasping motion is generated that not only allows to pick up the objects but also to operate them, such that the object can be functionally used. This functional requirement imposes several constraints on the manner how the object should be grasped and requires high accuracy in the final grasp pose. Fig. 4.1 shows a functional grasp transferred from a canonical model of the spray bottle category. The grasped instance has never seen by

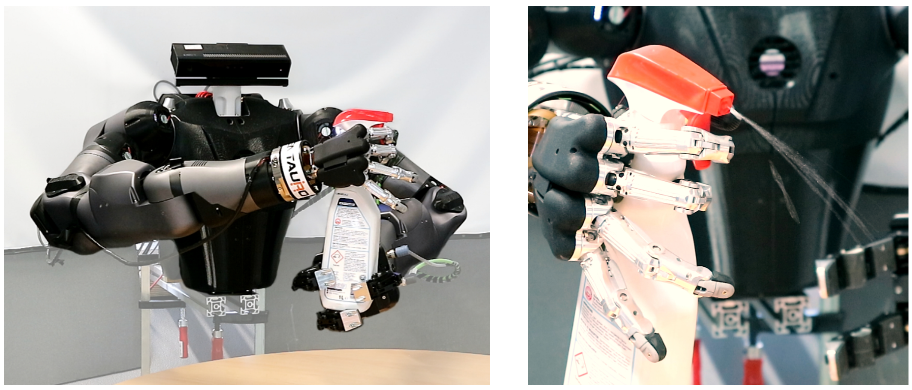
Figure 4.1: Left: Transferred functional grasp performed by the Centauro upperbody. Right: functional use of the spray bottle.

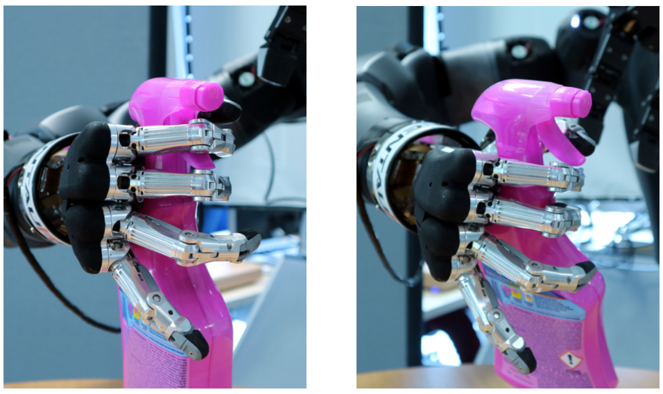
Figure 4.2: Functional (left) and non-functional (right) grasp of a spray bottle. Observe on the right that the trigger can not be activated which establishes the non-functionality of the grasp.

our method before. Furthermore, Fig. 4.2 presents a functional, and a non-functional grasp of a spray bottle. This Figure highlights the relevance and challenges of generating accurate grasps poses to enable the object usage.

The grasping motion is generated online, which is required for robotic applications in unstructured scenarios where objects to manipulate are unknown a priori. In addition, grasping objects that are only partially observed is another requirement for real robot applications. In this manner, when possible, regrasping and robot re-positioning operations, which incur in time and are pruned to errors, are avoided.

The grasping transfer has been implemented in several robotics applications, including single arm grasping, dual-arm (bimanual) grasping and regrasping. In the latter, the object pose does not allow a direct functional grasping and the object is firstly grasp with a supportive arm. A sample of a dual-arm grasp is displayed in Fig. 4.3.

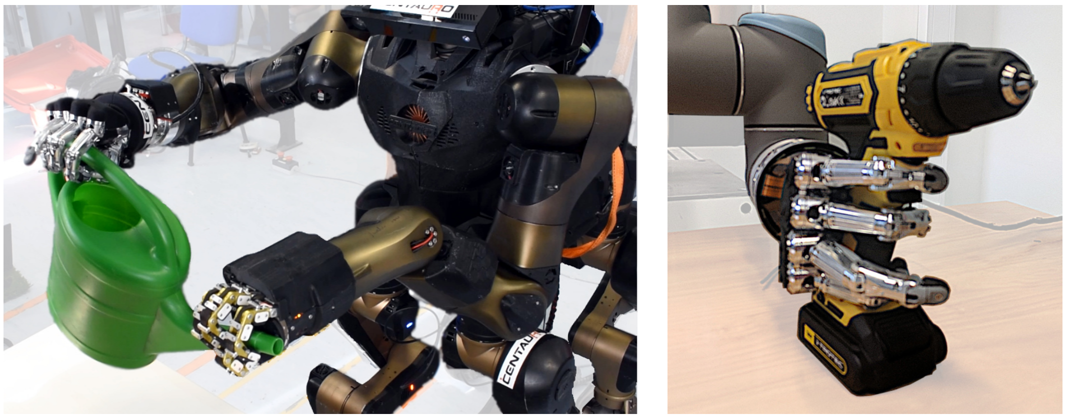

Figure 4.3: Left: Functional dual-arm grasp skill transfer of a watering can performed by the Centauro robot. Right: Functional grasp skill transfer of a drill by a UR10 manipulator. The multi-fingered SVH Schunk hand and four fingers Herii hand are attached at the end-effectors of the robot arms.

In general, the grasping transfer happens as follows. Given a canonical model with its respective grasping motion, described by a consecutive list of control 6D poses, an observed instance is non-rigidly registered which results in a dense deformation field. This deformation field finally warps the control poses of the canonical grasping motion delivering the final grasping motion. This grasping motion transfer is described in Section 4.2. Successful grasping experiences can be aggregated to enrich the canonical grasping motion which originally is defined by one grasping motion. This knowledge aggregation is described in Section 4.3. Finally, the skill transfer approach is evaluated, with and without knowledge aggregation, in Section 4.4.

The methods and experiments presented in this chapter has been published in [Pavlichenko et al., 2019; Pavlichenko et al., 2018; Rodriguez and Behnke, 2018; Rodriguez et al., 2018a]. The accompanying online Video 4.1 [1] shows the transfer process of some novel instances. Additionally, single and dual arm experiments performed with real platforms are presented, as result of the skill transfer.

4.1 Related Work

Several approaches have been proposed for robot grasping [Bohg et al., 2013]. Traditional analytical methods such as [Bicchi and Kumar, 2000; Huebner et al., 2009; Morales et al., 2006] aim to provide a guarantee criterion about the grasping quality. However, because of the lack of precise object models in real unknown scenarios, grasp synthesis approaches have been mostly investigated in simulated environments. Furthermore, several studies

[1] Video 4.1: `https://zenodo.org/record/3991986/files/chapter_4.mp4`

have pointed out the difficulties of employing such approaches in real applications, where such methods have under-performed significantly [Balasubramanian et al., 2012; Diankov, 2010].

Acquiring grasp skills by experience with real objects is then a promising alternative which does not require sim-to-real transfer. The main challenges are then translated to the object representation and the generalization to novel objects. The work presented here can be categorized as one of such approaches. We accumulate grasping experiences in learned models that are transferred to novel instances. In addition, we represent object geometries by means of low-dimensional shape spaces. This compact representation allows us to infer object descriptors in real time of novel instances, which is essential for grasp robot applications in unstructured scenarios.

The approach proposed by Stueckler et al. [2011] has served as source of inspiration for the approach presented in this chapter. In [Stueckler et al., 2011], a novel manipulation skill transfer method is introduced which is based on the non-rigid registration of two objects. The registration establishes correspondences between points belonging to the object surfaces and the grasp poses or motion trajectories are adapted according to these correspondences. However, in [Stueckler et al., 2011], both the canonical and the observed instances are known in before hand which restricts its application to unstructured real scenarios. To overcome this limitation, the approach presented here makes use of a lower-dimensional latent space which enables to formulate the non-rigid registration as a non-linear optimization problem capable of registering novel instances in an online manner.

Methods proposed in [Stouraitis et al., 2015] and in [Amor et al., 2012] also transfer control grasp poses according to a non-rigid registration that minimizes the euclidean distance between assigned correspondences. Interestingly, [Stouraitis et al., 2015] also handles explicitly functional power grasps. However, both [Stouraitis et al., 2015] and [Amor et al., 2012] are limited only to offline applications since these approaches are based on fully observed models. In contrast, the approach presented here is suitable for online scenarios.

To transfer grasping skills, Vahrenkamp et al. [2016] performs a part-wise segmentation of the objects based on their RGB-D appearance. Each of the parts has an associated template primitive which is matched to the observation. However, the primitives are not rich enough to represent all the expressiveness of the object models resulting in a poor performance on partial views.

None of previous approaches is able to aggregate grasping knowledge of previous successful grasping experiences. Kroemer et al. [2010] and Stulp et al. [2011] refine an initial grasping motion acquired by human demonstration to grasp objects in a reinforcement learning framework. Although both approaches are robust against noise on the object pose, they only handle known objects. In contrast, the approach presented in Section 4.3 is able to grasp novel objects belonging to a familiar category based on real noisy sensor data. Detry et al. [2012] proposed a grasp database that associates primitive shapes to grasp configurations. Parts of the observed instances are clustered such that each main part has an associated grasp. Although the use of grasp knowledge allows to perform grasps in real time, the objects are not reconstructed, and thus, functional grasps can not be guaranteed.

4.2 Skill Transfer

The grasp skill transfer is based on the non-rigid registration between on observed and a canonical instance. The registration can be performed following the method presented in chapter 2 or the approach detailed in chapter 3. Both methods deliver a dense deformation field in the manifold of the canonical object. As result of the registration, each 3D point of the canonical model is assigned to a 3D deformation vector. These deformation fields are then used to warp control poses of the grasping motion of the canonical model.

A grasping motion is defined as a sequence of parameterized motion primitives each of them described by a control 6D pose expressed in the object frame. Typical motions include a primitive to approach the object, characterized by a pregrasp pose, and a primitive to make contact with the object, characterized by a grasp pose. Each of the control poses of the motion are warped and the resulting poses define the transferred grasping motion. Naturally, motion primitives that do not affect the pose of the robotic manipulator are kept unmodified, e.g., a closing hand primitive motion.

Note that although several motions can pick up the objects, only few are able to grasp them in a functional manner. In the approach presented here, it is assumed that the motion of the canonical model is functional.

The warping is similar to the morphing process, the canonical model undergoes to deliver the reconstructed model of the observed instance. In this manner, the Coherent Point Drift (CPD) equations are employed to warp the control poses. Recalling Eq. (2.2) and Eq. (2.4), a deformed instance is expressed as:

$$\mathcal{T} = \mathbf{S}^{[t]} + G(\mathbf{S}^{[\mathbf{t}]}, \mathbf{Z})\mathbf{W}. \tag{4.1}$$

During shape inference, the moving points $\mathbf{S}^{[t]}$ are replaced by the canonical model $\mathbf{C}$. The values of the Gaussian kernel matrix $G(\mathbf{S}^{[\mathbf{t}]}, \mathbf{Z})$ can be interpreted as weights between points according to the distance to the base points $\mathbf{Z} = \mathbf{C}$. For warping a control frame $\mathbf{B}$, Eq. (4.1) is employed setting the moving points $\mathbf{S}^{[t]}$ to the origin $\mathbf{T_B}$ of frame $\mathbf{B}$, or equivalently to its translation component. In this manner, $G(\mathbf{B}, \mathbf{C})$ establishes weights to each point of the canonical model to the origin of $\mathbf{B}$ according to their distance. These weights are multiplied by the inferred deformation field $\mathbf{W}^*$ of the observed instance to obtain the deformed origin $\mathbf{T_B}'$ of the origin of $\mathbf{T_B}$:

$$\mathbf{T}'_\mathbf{B} = \mathbf{T_B} + G(\mathbf{T_B}, \mathbf{C})\mathbf{W}^*. \tag{4.2}$$

Warping the rotation $\mathbf{R_T}$ of frame $\mathbf{B}$ is more elaborated because the deformation process does not have any notion of rotational constraints, e.g., the orthonormality of a

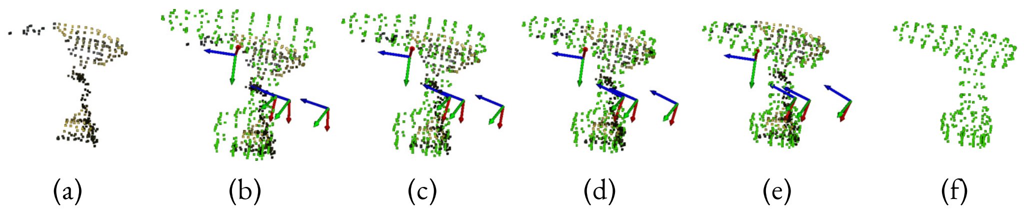

Figure 4.4: Grasp transfer of a partially observed real drill. The green point cloud represents the canonical instance. The input point set is displayed on (a). Deformation of the canonical model is presented from (b) to (e). Note how the control grasp poses are deformed correspondingly. The reconstructed model is finally shown on (f).

rotation matrix or the relation between the scalar and the vector part for quaternions. The deformation of the rotation is approximated by warping each of the vectors belonging to the rotational base $\mathbf{R}$ and orthonormalizing the resulting warped vectors to fulfill the rotation constraints. Each vector of $\mathbf{R}$ is scaled by $\epsilon = 1 \times 10^{-4}$ and added to $\mathbf{T_B}$. These points are warped and the deformed origin $\mathbf{T_B}'$ is subtracted from each of them to generate three vectors corresponding to a non-orthonormal base. Finally, this base is orthonormalized by Gram Schmidt to generate a valid rotation matrix $\mathbf{R_T}'$.

To transfer a grasping motion from a canonical to an observed instance, each control pose of the motion primitives is warped as described above. Fig. 4.4 shows the warping process of three control poses that define the grasping motion for the drill category. The input comes from a real RGB-D camera observing the object from a single view.

4.3 Grasp Skill Aggregation

The grasp skill transfer presented in previous section is based on the grasping motion of the canonical model. This is in general a handcrafted motion that guarantees a successful grasping of the canonical model. However, in case other instances of the same category have also been successfully grasped, this knowledge is not being incorporated in the grasp transfer. Therefore, in this section, an approach to aggregate knowledge of successful grasping motions of different instances into a skill transfer model is presented.

The aggregation process transforms a descriptor ς_i of a grasping motion of an object instance $\mathbf{T}_i$ into a common object space, e.g., into the object space of the canonical model. Fig. 4.5 shows two object instances with their respective control grasping poses transferred to the object space of the canonical model. Additionally, a shape descriptor $\mathbf{x}_i$ of $\mathbf{T}_i$ is inferred and together with the grasping descriptor ς_i define a training sample of the skill transfer model. An overview of the training of the grasp skill transfer model is depicted in Fig. 4.6. The shape space is constructed as stated in Section 2.3.

The transformation of a 3D point $\mathbf{o}$ expressed in an observed instance space into the canonical space is defined by finding the inverse transformation $v^{-1}(\mathbf{o})$ of Eq. (2.4) which

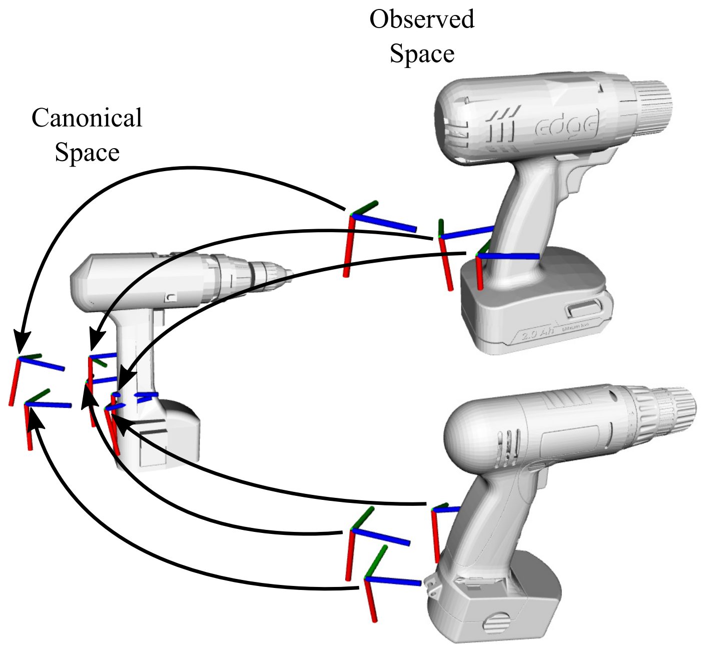

Figure 4.5: Grasping skill aggregation. The control poses of the grasping motions of two observed instances are transformed into the object space of the canonical instance by Eq. (4.3).

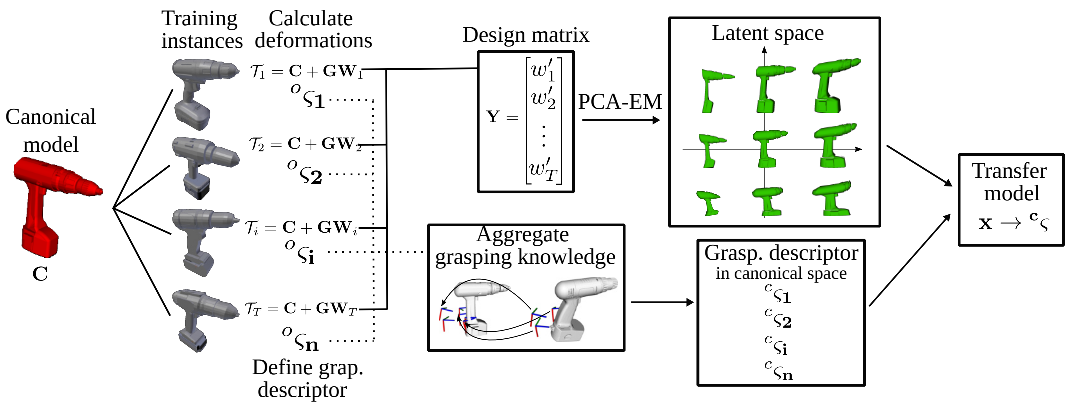
Canonical model
$\mathbf{C}$
Training instances
Calculate deformations
$\mathcal{T}_1 = \mathbf{C} + \mathbf{G}\mathbf{W}_1$
${}^{O}\boldsymbol{\varsigma}_\mathbf{1}$
$\mathcal{T}_2 = \mathbf{C} + \mathbf{G}\mathbf{W}_2$
${}^{O}\boldsymbol{\varsigma}_\mathbf{2}$
$\mathcal{T}_i = \mathbf{C} + \mathbf{G}\mathbf{W}_i$
${}^{O}\boldsymbol{\varsigma}_\mathbf{i}$
$\mathcal{T}_T = \mathbf{C} + \mathbf{G}\mathbf{W}_T$
${}^{O}\boldsymbol{\varsigma}_\mathbf{n}$
Define grap. descriptor
Design matrix
$\mathbf{Y} = \begin{bmatrix} w'_1 \\ w'_2 \\ \vdots \\ w'_T \end{bmatrix}$
PCA-EM
Latent space
Aggregate grasping knowledge
Grasp. descriptor in canonical space
${}^{c}\boldsymbol{\varsigma}_\mathbf{1}$
${}^{c}\boldsymbol{\varsigma}_\mathbf{2}$
${}^{c}\boldsymbol{\varsigma}_\mathbf{i}$
${}^{c}\boldsymbol{\varsigma}_\mathbf{n}$
Transfer model
$\mathbf{x} \rightarrow {}^{\mathbf{c}}\boldsymbol{\varsigma}$

Figure 4.6: Training phase of the grasping skill transfer model. The shape space of an object category is constructed as explained in Section 2.3. The control poses of the grasping motion for each training sample are transformed in the canonical object space to be aggregated. The shape descriptor $\mathbf{x}$ serve as feature vector while the grasping descriptor ${}^{C}\boldsymbol{\varsigma}$ is the label for the grasping transfer model.

maps 3D points from the canonical to the instance space. This inverse transformation is estimated by a distance weighted interpolation of a set of points $\mathbf{Z} = \{\mathbf{z}_1, \cdots, \mathbf{z}_M\}$ around the point $\mathbf{o}$ in the observed space to be transformed, namely:

$$v^{-1}(\mathbf{o}) = -\frac{\sum_{i=1}^{M} G(\mathbf{o}, \mathbf{z}_i + v(\mathbf{z}_i))v(\mathbf{z}_i)}{\sum_{i=1}^{M} G(\mathbf{o}, \mathbf{z}_i + v(\mathbf{z}_i))}. \tag{4.3}$$

Note that the Gaussian kernel $G(\mathbf{o}, \mathbf{z}_i + v(\mathbf{z}_i))$ establishes the contribution of a point $\mathbf{z}_i$ around $\mathbf{o}$. This contribution is set according to the distance from $\mathbf{z}_i$ to $\mathbf{o}$.

Grasping poses in the observed space are transformed by warping the origin and the rotational base as explained in Section 4.2. The transformed poses are then expressed as grasping descriptors $\boldsymbol{\varsigma}$ composed of the translation vector and the rotation following a roll, pitch and yaw convention. This particular rotation representation is chosen over quaternions and rotation matrix because of the difficulties of traditional supervised learning methods to incorporate rotation constraints such as orthonormality and the scalar-vector relation for quaternions.

The grasp skill transfer model is trained as follows. Given a set of 3D models with a corresponding functional grasping motion for each of the instances in the set, a shape space is built as explained in Section 2.3. A shape descriptor $\mathbf{x}_i$ is inferred for each of 3D model of belonging to the training set. Then, the control poses of the grasping motions of each instance are transformed into the space of the canonical instances as described above, resulting in a grasping descriptor $\boldsymbol{\varsigma}_i$ for each instance of the training set. Each shape vector represents the features and its associated grasping descriptor the target of a training sample for the grasping transfer model. A simple linear regressor constitutes the transfer model.

The grasping motions of the observed instances can be set manually or can be automatically generated in order to reduce time and wear off of the real robotic system. The automatic grasping motion generation is a sampled-based method by generating 6D random poses around the control poses of the grasping motion of the canonical instance. For the translation, a three-dimensional vector is sampled from a multi-normal distribution with mean equal to the origin of the control pose and a given standard deviation σ_T. For the rotation, a quaternion is sampled following the approach proposed in [Shoemake, 1992] with a given σ_R. The sampled control poses are then filtered such that a functional grasp is ensured. This constraints depends on the object category. For the drill category, for instance, poses that impede that use of the trigger are discarded. Generated motions that result in collisions are also filtered out. Fig. 4.7 illustrates three sampled motions for a drill instance. Finally, the sampled motions are executed in simulation. If the objects are functionally grasped, the grasping motion is assigned to the corresponding instance.

Geometrical primitives (spheres, cylinders, boxes and capsules) can be used in order to optimize the simulation time, since collision checking involve expensive operations. For this purpose, an automatic primitive generator is implemented based on the Roboptim library [Khoury et al., 2013], that calculates optimal oriented capsules for 3D meshes. Our

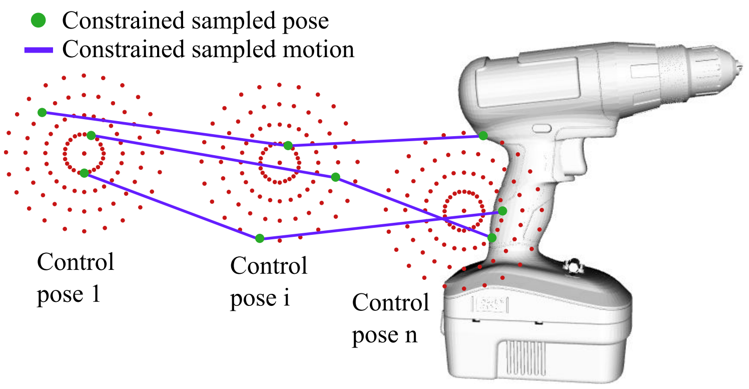

Figure 4.7: Sampled-based grasping motion generation. Control poses are sampled from the canonical grasping motion and functionally evaluated. The red points illustrates the origin of the sampled control poses. For simplicity, a 2D representation is shown.

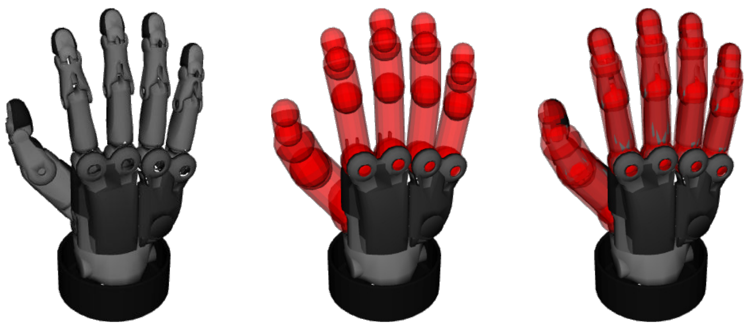

Figure 4.8: Collision model of the Schunk SVH hand approximated by geometrical primitives. Left: Detailed 3D model of the robotic hand. Center: fingers modeled as capsules. Right: visual and collision model of the robotic hand to show the quality of the capsule approximation.

implementation is ROS compatible and is open-sourced [2]. An example of the primitives generated for the multi-fingered Shunk hand employed for the experiments is shown in Fig. 4.8.

In inference, the problem consists of finding a functional grasping motion for a novel presented instance. In other words, given a shape descriptor $\mathbf{x}^*$ of the instance, infer a grasping descriptor ${}^{O}\varsigma$. The shape is inferred by Eq. (2.11) as explained in Section. 2.3 which finds a latent vector $\mathbf{x}^*$ and a local rigid registration $\boldsymbol{\theta}^*$. The grasping descriptor is inferred in three steps. First, a grasping descriptor ${}^{C}\varsigma$ expressed in the space of the canonical model is inferred by the skill transfer model, i.e., the linear regressor. Second, ${}^{C}\varsigma$ is transformed into the space of the observed instance by warping all control poses as explained in Section. 4.2. Finally, the resulting grasping descriptor ${}^{O}\varsigma$ is transformed by

[2] `https://github.com/AIS-Bonn/primitive_fitter.git`

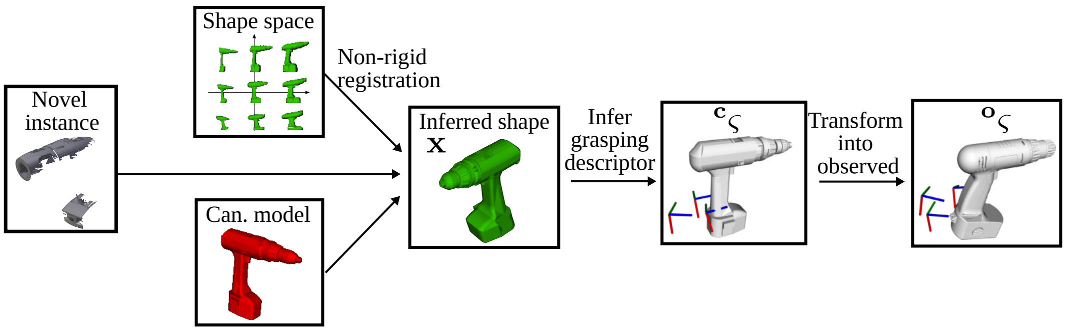

Figure 4.9: Inference of functional grasping motions given the observed shape. A canonical model (red) is deformed to match a partially-occluded instance (leftmost object) by a shape space registration. The grasping descriptor $^{c}\varsigma$ is inferred from $\mathbf{x}$ and transformed into the observed space.

the inferred local registration $\boldsymbol{\theta}^{*}$. This rigid transformation makes our approach more robust against misalignments coming from pose estimation modules. The inference phase is illustrated in Fig. 4.9.

4.4 Evaluation

The grasp skill transfer has been applied in simulation and real scenarios with single-arm and dual-arm robot manipulators. The approach presented here is agnostic to the kinematics of the manipulator, it is only assumed that the robot arm is able to reach 6D poses inside the workspace. Our approach does not model the motion of robotic hands explicitly, this is discussed in Chapter 5. It is assumed for the evaluation presented in this section that a functional grasp depends only on the approach and final grasp configuration. The experiments were carried out with three robotic platforms: a UR10 with an attached Schunk right hand [Ruehl et al., 2014] as end-effector, a Centauro robot [Klamt et al., 2018] with a right Schunk hand and a four DOF Herii hand [Ren et al., 2017], and a Centauro upperbody with a right Schunk hand and a four DOF Robotiq gripper. The UR10 has 6 DOFs while the Centauro robot has two manipulators with 7 DOFs each. Both the UR10 and the Centauro robot are shown in Fig. 4.3. The Centauro upperbody with the Robotiq hand is presented in Fig. 4.1. The Schunk hand has 20 joints including 11 mimic ones which results in 9 DOFs (Fig. 4.8).

4.4.1 Single-arm Experiments

Initial experiments were carried out for the drill category with 10 training instances and the UR10 robotic platform. The transfer was tested both in simulation and in real environments. The simulated tasks were implemented in Gazebo [3]. For collision detection, the link meshes were replaced by geometrical primitives (Fig. 4.8). Cross-validation was employed due to the reduced number of objects. Five shape spaces were trained leaving two instances for testing for each of the latent spaces with $\beta = 1$ and $\lambda = 3$ for the CPD computations and 5 latent dimensions. A grasp was successful if the drill was held by the hand after lifting it. A success rate of 80% was achieved in this test. Failed grasps were attributed mainly to the size of the testing objects compared with the training instances when the biggest or smallest drills were chosen as training instances. This indicates that our approach can benefit from data augmentation techniques. In real environments, we perceived three novel drills by integrating a Kinect v2 sensor [Fankhauser et al., 2015] into the robotic system. The shape space was built with all 10 training instances. All three real instances were successfully grasped. The inference in average took 6 s on a Corei7 CPU 2.6 GHz, which demonstrates that our method can be implemented on-board in real robots. A grasp transfer from one of the real testing instances is presented in Fig. 4.4.

The grasp skill aggregation approach (Section 4.3) was also evaluated in simulation and with a real robotic platform. The method was evaluated on the drill and spray bottle categories, each of them containing 13 and 17 instances, respectively. The experiments were carried out in the Gazebo simulator. The shape spaces were parameterized as the previous experiment described above. For building the grasping transfer models, motions were sampled with $\sigma_T = 0.04$m and $\sigma_R = 0.2$ for sampling the translation an the rotation components, respectively. Due to the reduced number of training instances, we use cross validation building six (drill) and seven (spray bottle) grasp transfer models, leaving two instances for testing. Our method is evaluated with fully and partially observed objects represented as point clouds. The partially observed point clouds are generated by ray-casting from a single view. Examples of inferred grasping motions and model reconstructions are shown in Fig, 4.10.

Results of the grasp transfer by knowledge aggregation are presented in Table 4.1. In total, 52 grasping motions were transferred, since each of the 13 grasping transfer models was evaluated on two fully, and on two partially observed instances. In total, 30 motions lead to a successful grasp which corresponds to a 57.7%. Several failures are attributed to collisions with the object while reaching the approach pose. Collisions

Table 4.1: Ratio of successfully transferred grasps of the aggregated grasp transfer approach.

	Drill		Spray Bottle	
	Grasp	Func. Grasp	Grasp	Func. Grasp
Fully observed	7/12	4/12	8/14	3/14
Partially observed	6/12	3/12	9/14	6/14

[3] http://gazebosim.org/

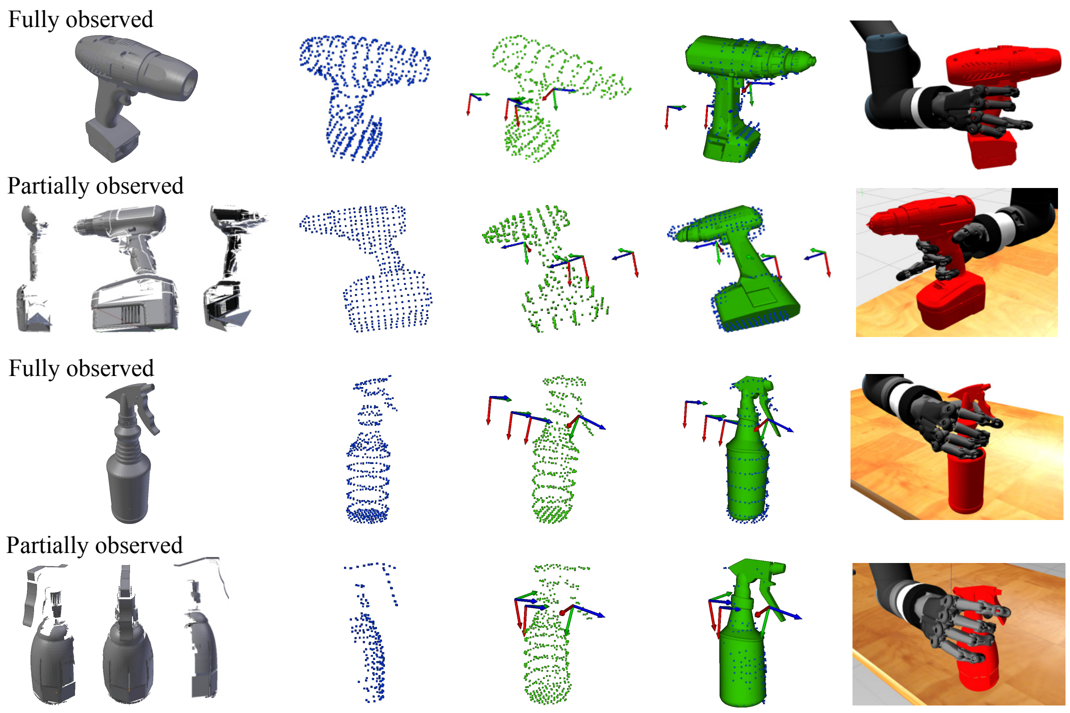

Figure 4.10: Evaluation of the grasping skill aggregation in simulation. Fully and partially observed instances of the drill and spray bottle categories are grasped. Meshes are depicted at the leftmost column. The single partial views are shown from three different perspectives. The corresponding point clouds of the meshes are colored blue. The reconstructed instances (point clouds and meshes) are displayed in green. To evaluate the quality of the registration, the ground truth object model is presented with the respective inferred point cloud. The resulting grasped object is shown at the rightmost column.

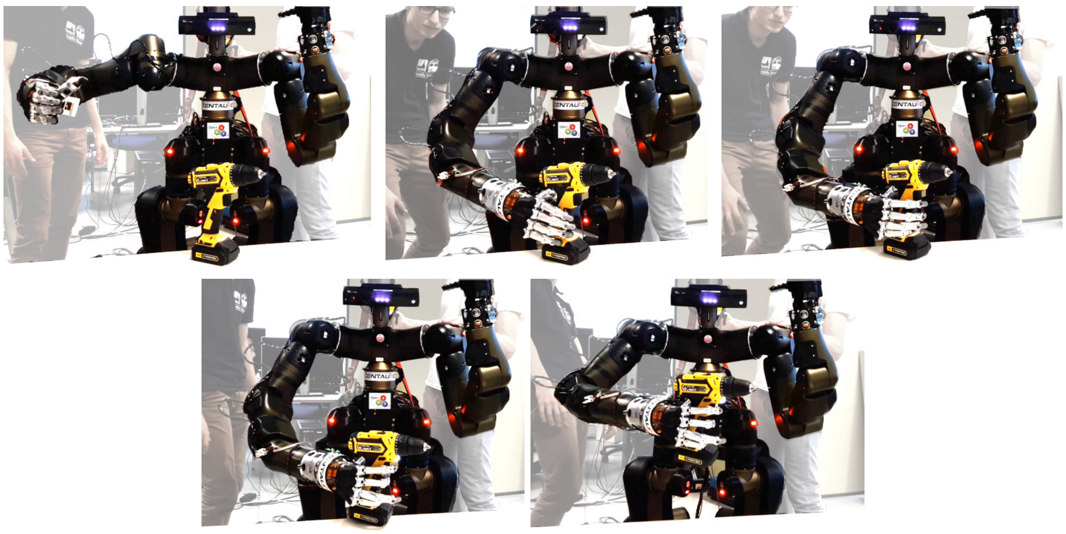

Figure 4.11: Centauro robot grasping autonomously a drill presented by the first time. A functional grasp is successfully transferred that allows the object to be used.

with the environment also caused some failures. Considering parts of the warped model as obstacles and incorporating path planning methods in our approach is a promising alternative to solve these issues. In average the transfer including the registration, took 7s which demonstrates the applicability of the method on online scenarios. A successful grasp of a drill instance with the real Centauro was also achieved. The input was a point cloud of the objects coming from single-view captured by the Kinect v2 sensor. Snapshots of the grasping motion are presented in Fig. 4.11.

4.4.2 Dual-arm Experiments

Dual-arm grasping imposes a harder problem for the grasp planning, that requires a higher level of accuracy. Undesired contacts of one arm with the object might affect the estimated pose leading to a non-functional grasp or to a grasp failure in the worst case. The main difference with single-arm grasping is the simultaneous warping of multiple control poses. In this manner, each manipulator defines an individual grasping motion composed of primitives which are characterized by control poses. In dual-arm applications, those poses are transferred simultaneously. Both arm motions are executed synchronous to obtain the final grasping motion. The experiments are performed with the Centauro robot, but only the upperbody was actuated.

Initial experiments were carried out in the Gazebo simulator for the watering can category. For the evaluation, 8 training instances were selected to train the shape space of the category and three instances were left for testing. A shape space with 8 latent dimensions was constructed with the training instances. The testing objects were perceived

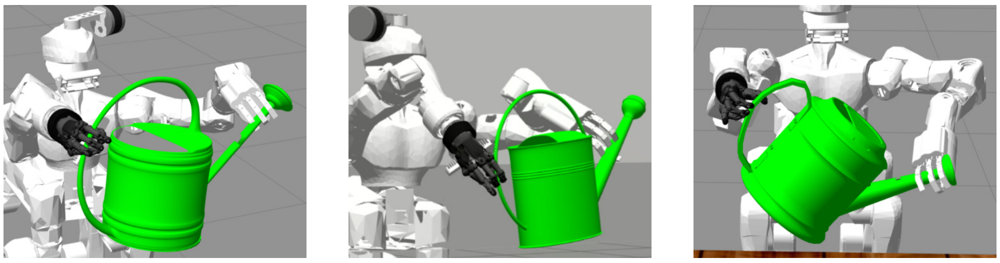
(a) Can 1 (b) Can 2 (c) Can 3

Figure 4.12: Dual-arm grasp transfer of watering cans in simulation. All of testing instances are different and presented by the first time.

Table 4.2: Success rate of picking watering cans in simulation in dual-arm grasp tasks.

	Success rate (attempts to solve)
Can 1	75% (4)
Can 2	100% (5)
Can 3	60% (5)

by a simulated RGB-D sensor. Successful grasps of the three testing objects are shown in Fig. 4.12. Each can is placed on a table in front of the robot with three different rotations along the vertical axis: 0, -0.25, and 0.25 radians. A trial was considered as success if the watering can was grasped and lifted three times, one for each orientation. Success rates are presented in Table 4.2. The can with lower success rate possesses the most distinctive geometry compared to the instances in the training set which suggests an increment in the variability of the geometries in the training objects may improve the results of the transfer.

For dual-arm grasping, real robot experiments were also performed. The experiment consisted of lifting a watering can from a single orientation with the real Centauro robot. From five attempts, a functional grasp was transferred four times successfully. The average inference time was $4,51 \pm 0.69$ seconds. Additionally, a bimanual grasp of a drill with a handle was carried out with the Centauro robot. Both instances, the watering can and the handle drill, were completely novel when presented to our approach. Snapshots of the dual-arm grasping process are shown in Fig. 4.13.

4.4.3 Regrasping Experiments

The feasibility of a functional grasp depends on the object pose. Sometimes, objects are placed such that a direct functional grasp is not possible. For example, when the trigger of a drill or spray bottle has contact with the environment impeding a direct object grasp

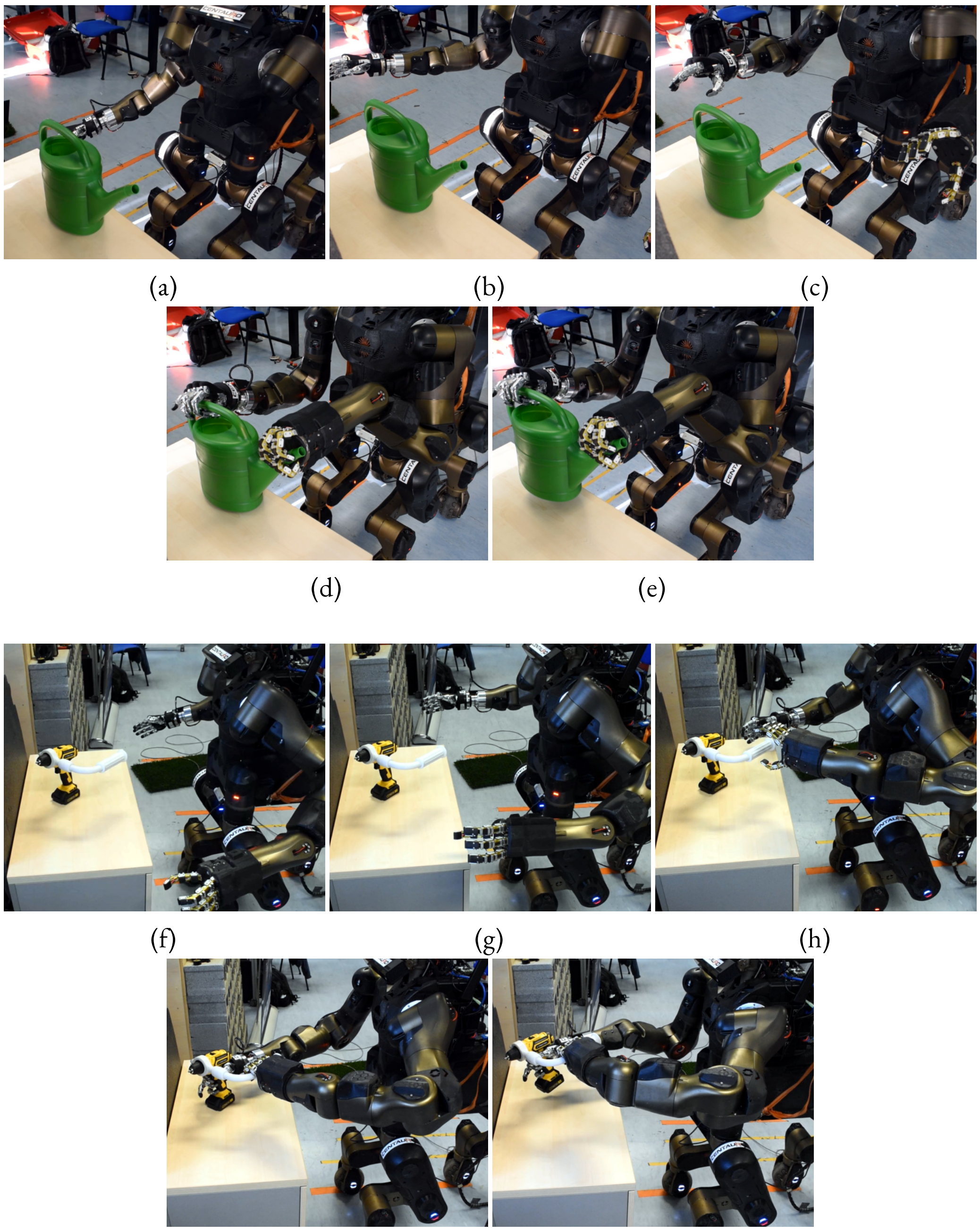

Figure 4.13: Dual-arm grasping with the Centauro robot of a watering can and a handle drill. **(a)** and **(f)** Initial poses. **(b-c)** and **(g-h)** Reaching the pre-grasp pose. **(d)** and **(i)** Object is grasped. **(e)** and **(j)** Can/drill is lifted.

Table 4.3: Success rate of regrasping tasks

Category	Object	Func. Grasp	Grasp	Trials
Spray Bottle	White	8	9	9
	Blue	6	7	9
	Pink	5	8	9
	OVERALL	19 (70%)	24 (88%)	27
Watering Can	Dark green	4	4	6
	Light green	6	6	10
	Yellow	7	7	10
	OVERALL	17 (65%)	17 (65%)	26

without collisions. In these situations, regrasping strategies are required. Inspired by the anthropomorphic nature of human body, we proposed an approach that performs an initial grasp with one hand obtaining full control of the object pose and allowing a functional grasp with a second hand.

In addition to the grasp transfer, regrasping requires refinement of the object pose because of possible misalignments caused by the initial grasp. This refinement is based on the Iterative Closest Point (ICP) method given a filtered point cloud of the object in hand and the reconstructed model coming from the shape registration module. The refined pose is then employed to update the inferred grasp poses.

The grasp transfer presented in this chapter has been also evaluated for regrasping tasks. For the evaluation, the Centauro upperbody introduced above is employed. Three instances for each of the watering can and spray bottles categories are placed in front of the robot on a table such that a direct grasp can not be performed because of collisions with the environment or kinematics constraints. The task consisted in grasping initially the object with the left hand in order to perform a second functional grasp with the right hand. A shape space was trained for each of the categories with 16 an 8 training instances, respectively.

Each testing instance of the spray bottle is placed lying in three different orientations: $-45°$, $0°$, and $45°$. For each object orientation, three grasping trials are attempted resulting in 27 grasp attempts. Success rate of the regrasping task are presented in Table 4.3. Note that failures do not only depend on the grasp skill transfer but they might come from other modules such as the ICP-based in-hand pose refinement. In fact, in the three cases, where even a non-functional grasp was not achieved for the spray bottles, failures were caused by the ICP registration and by a handover planner (not described in this thesis). Failure cases, however, in which a grasp was possible for the spray bottles but the functionality was not guaranteed, were mostly caused by a error in the mesh registration which lead to

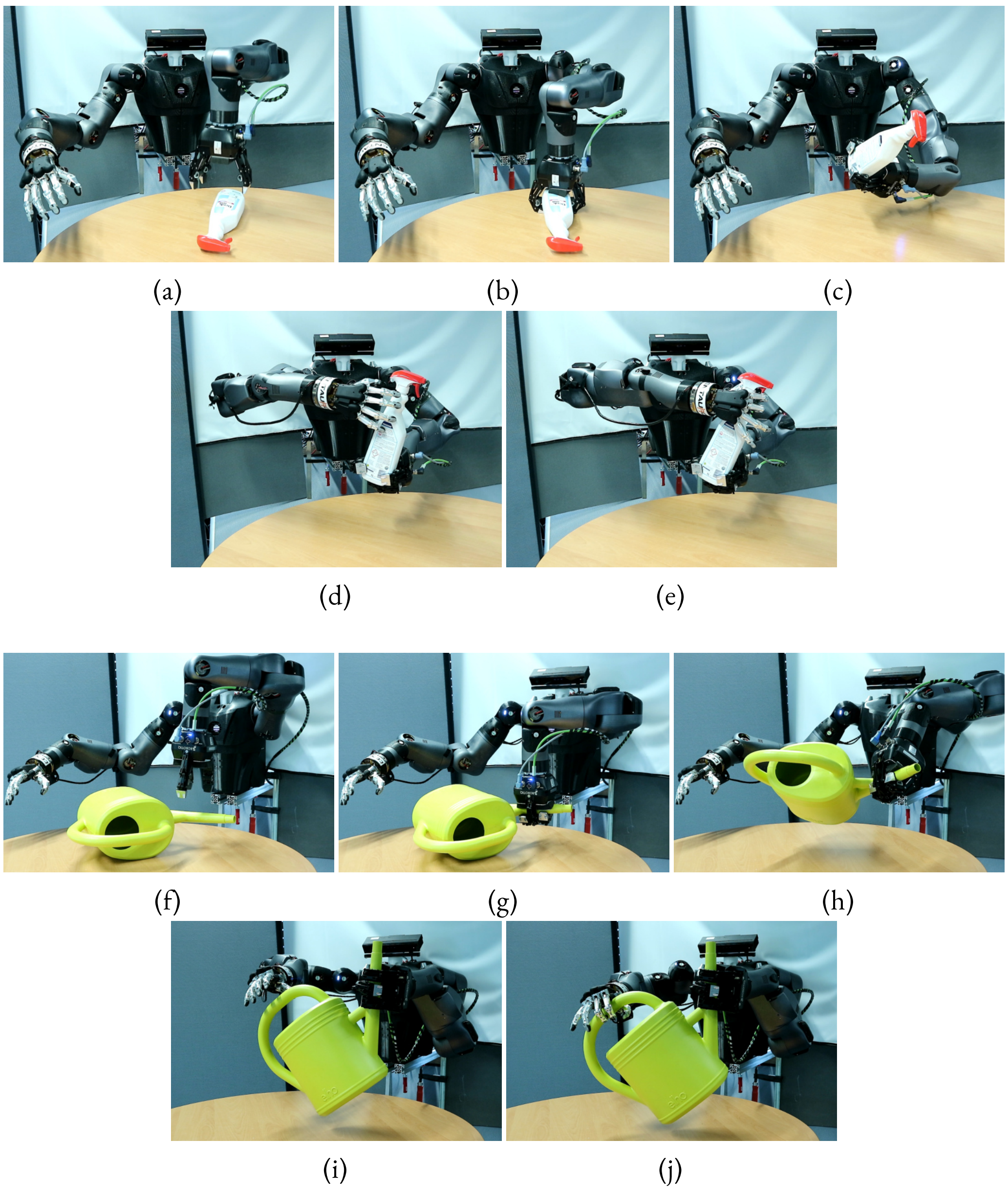

Figure 4.14: Functional regrasp of a spray bottle and a watering can. **(a),(f)** Pregrasp pose of the supportive grasp. **(b),(g)** Supportive grasp with left hand. Observed how the object pose is changed because of the initial grasp. **(c), (h)** In-hand object pose estimation that corrects deviations introduced by the initial grasp. **(d-e), (i-j)** Functional grasp.

a wrong grasp transfer. Regarding the watering cans, the problems were caused mainly by the initial grasp with the left hand which was not modeled or transfer by the method presented in this chapter. Only once, a faulty mesh registration caused a failure regrasp of the object. Successful examples of regrasping operations for a spray bottle and a watering can are shown in Fig. 4.14. The runtime of the shape registration together with the grasp transfer took in average 8.59 ± 0.88 for the spray bottle category and 15.83 ± 0.60 for the watering can category. This demonstrates again that this method can be applied online.

4.5 Discussion

The shape space registration method introduced in Chapter 2 was designed to consider the requirements for skill transfer. In this chapter, a grasp skill transfer approach was presented based on the object geometry of the observed instance. This geometry is represented by means of a shape space descriptor. The transfer is described by warping control poses of parameterized grasping motion priors. The rotation of the warped control poses is orthonotmalized to meet the constraints of a valid rotation matrix. Our approach does not require to know the 3D model of the observed instance and is able to reconstruct the object shape.

Because of the expressiveness of this approach, it has been implemented to solve multiple single-arm and dual-arm grasping tasks. Our approach has been applied to different categories including drills, spray bottles, watering cans and drills with handles. This demonstrates the applicability of the method to different object categories. Moreover, the work presented in this chapter generates functional grasps which enables the object usage. The evaluation has shown that this method can be applied online with noisy input data captured by common RGB-D sensors.

5 Learning Postural Synergies for Category-level Grasping by Shape Space Registration

"Admiration for the beauty of and belief in the logical simplicity of the order and harmony which we can grasp humbly and only imperfectly."
— Albert Einstein

In the previous chapter, grasping motions were transferred to novel instances based on the shape of the objects. The motion was represented as a series of control poses that are warped using a dense deformation field obtained from the non-rigid registration between a canonical and an observed instance. It has been assumed so far, however, that the entire grasping process depends only on the approaching and goal poses of the end effector. In this manner, the motions of the robotic hands were fixed for all instances of the same category. This assumption strongly depends on the controller of the robotic hand. For example, the implementation of a simple position controller combined with that strategy might damage the robotic hand or generate very loose grasps, depending on the object shape and material. Thus, the controller implemented in the experiments of previous chapter was a position-current cascade controller, in which the hand is commanded to track a desired position without violating current thresholds. This provides compliance to the hand motions and allows the hand to adapt to the shape of the instances.

For object categories whose variation in size is significant, e.g., for a sphere category that contains instances with small and large diameters, a fixed hand motion combined with a position-current controller does not suffice. Objects with different sizes require different hand configurations. This chapter addresses this issue. Based on the shape representation of the instances, a hand configuration is inferred that adapts the hand joint configuration to the geometry of the objects. The object shape is modeled using the shape space described in Chapter 2, while the hand configuration is represented through postural synergies. The shape and synergy spaces need to be trained beforehand, giving our approach a data-driven nature. In inference, both learned models are exploited to infer the grasp hand configurations.

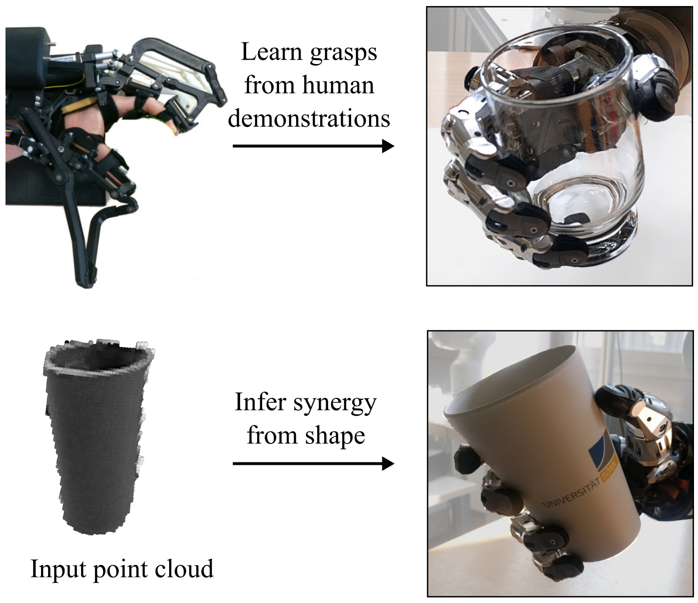

Figure 5.1: Grasping data is captured by an exoskeleton through human demonstration. Hand configurations are then inferred based on the captured data encoded in the synergy space of the robotic hand and the shape of the object.

The synergy space is a low-dimensional representation of possible hand configurations. This space is constructed from real hand configurations of the robotic hand following a grasp taxonomy [Feix et al., 2016]. In contrast with mappings from other robotic platforms or human grasping postural synergies, the synergy space is built with data of the real robot directly via human demonstrations. In this manner, approximation errors and overload of (e.g., human) kinematic adaptations are reduced. Errors stemming from visual systems and the corresponding camera calibration are also avoided. The acquisition of enough demonstrations with the robotic hand by commanding single joints is, however, very time consuming. Thus, an exoskeleton is employed for capturing the data for building the synergy space. Apart from controlling the robotic hand in an intuitive and fast manner, the exoskeleton provides force feedback useful for assessing the grasp quality. Fig. 5.1 shows the exoskeleton that captures grasping data through human demonstration, and an inferred hand configuration given an object observation.

The approach presented here can be implemented online in real robotic systems. Given a shape description, the inference time takes less than a second. So, the total time is mostly dominated by the shape descriptor inference that takes less than 10 seconds on average. As input, only a point cloud of the observed object is required. The objects can be partially observed by common 3D sensors. The instances are reconstructed thanks to the embedded knowledge of the shape space.

This chapter gives the argumentation published in [Rodriguez et al., 2018c]. An overview of the method with media material is available online in Video 5.1 [1]. Real robot grasping experiments containing a complete pipeline for approaching the object are also shown in the video.

5.1 Related Work

In the grasping community, postural synergies are usually employed to reduce the dimensionality of multi-fingered hands with high Degrees of Freedom (DOF) [Bernardino et al., 2013; Ciocarlie and Allen, 2009; Ficuciello et al., 2016a; Ficuciello et al., 2016b]. Synergies are mainly used for grasp planning and control. To construct synergy spaces, an anthropomorphic taxonomy is normally followed [Feix et al., 2016]. For the data acquisition necessary for building the synergy spaces, different methods have been proposed. Amor et al. [2012] and Ficuciello et al. [2016b], for example, integrated visual sensory data with kinematic mappings. The former captures human hands grasping objects and the latter transfers grasps from joint spaces of other robotic hands. These approaches, however, suffer from visual system errors and human-to-robot mapping inaccuracies. This issue was addressed by Bernardino et al. [2013] by measuring directly the joint position of the robotic hand through a data glove. We go a step further, and in the approach presented in this chapter, force feedback is provided to the operator to assess the grasp quality. Therefore, the data acquisition is verified both by visual, and by force feedback.

Through human demonstration acquired by data gloves, Ekvall and Kragic [2007] are able to infer pregrasp vectors based on shape primitives such as spheres, boxes and cylinders. Also by using synergies, Ficuciello et al. [2016a] modify postural synergies in a reinforcement learning problem that optimizes a force-closure reward function. Later, Ficuciello et al. [2016b] infers synergy values from basic geometrical parameters including height, length and diameter. The inference is carried out by training a fully connected neural network. In contrast to these previous approaches, a more complex and informative shape descriptor is autonomously inferred through our shape space registration. An interesting work to compare our approach was published by Faria et al. [2014]. There, superquadrics are employed to represent segmented point clouds of object parts. Synergy vectors are then inferred by training a Bayesian model. Unlike [Faria et al., 2014], our approach allows a functional grasp and the objects are reconstructed by the category-like information encoded in the shape space.

[1] Video 5.1: `https://zenodo.org/record/3991986/files/chapter_5.mp4`

5.2 Postural Synergies

The high number of DOFs of a multi-finger robotic hand is described by a low-dimensional vector residing in a latent space called the synergy space. The kinematics and model of each robotic hand establishes a unique synergy space. In this chapter, the anthropomorphic Schunk robotic hand is used as platform. The hand possesses 20 joints, however only 9 are fully actuated, the resulting 11 joints are coupled. The thumb has 2 DOFs for flexion and abduction movements. The index and middle finger are actuated in the proximal and medial joints, the distal is a mimic joint of the medial. The ring finger and the pinky only have actuation at the proximal joint, the medial and distal joints are coupled. The remaining actuated joint moves all fingers simultaneously in the spread direction.

Having only one actuator for spreading reduces complexity but imposes constraints on the Cartesian position of the fingers. This implies that there is no guarantee that 4 position targets for the index, middle, ring and pinky fingers, can be achieved at the same time, i.e., fulfilling one position target pose might cause a violation of the other three. This constraint and the lack of a medial actuated joint for the ring and little fingers, and the absence of a distal actuated joint for the thumb, renders the human-to-robot kinematics problem as undetermined. Such a mapping will also not exploit all capabilities of the robotic hand. Hence, the synergy space of the Schunk hand is found by grasping directly objects with the robotic hand instead of using any human-to-robot mapping.

The control of a multi-finger robotic hand requires the simultaneous coordination of several joints. This simultaneous control is tedious by motion editors that modify individual joint positions. An exoskeleton is thus employed to teleoperate the robotic hand

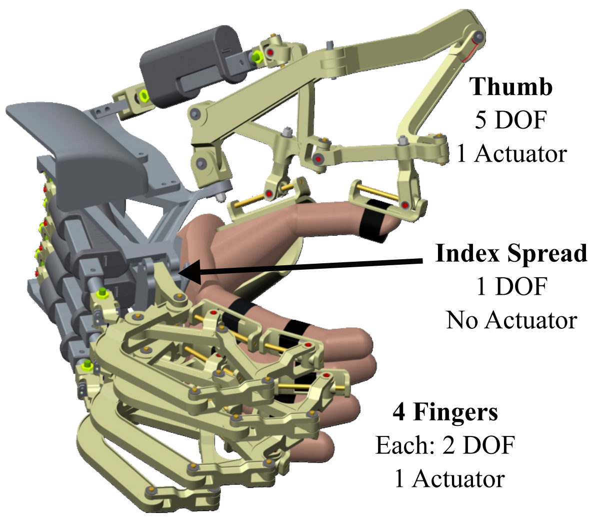

Figure 5.2: The U-HEx exoskeleton that teleoperates the robot hand. The exoskeleton can observe 14 DOFs of human motions and provides kinesthetic feedback to the operator.

in a natural manner exploiting the symmetry between the human and the robotic hand. Specifically, the U-HEx underactuated hand exoskeleton developed by the PERCRO lab of Scuola Superiore Sant'Anna is used to operate the robotic hand (Fig. 5.2).

The exoskeleton contains five independent parallel kinematics attached to a base that is fixed to the operator hand-back as shown in Fig. 5.2. The thumb exoskeleton is able to measure the configuration of the finger using 5 DOFs, which allows to fully identify the human movements [Gabardi et al., 2018]. The other four fingers have two DOFs that observe the *metacarpal-proximal joint* (MCP) and the *proximal-interphalanx joint* (PIP) [Sarac et al., 2017]. In addition, an extra DOF measures the spread movements of the index finger. The exoskeleton has five actuators to provide force feedback to the operator. This kinesthetic forces are transmitted through two human-exoskeleton contact points. An example of a three finger precision grasp operated by the exoskeleton is depicted in Fig. 5.3. Observe how the operator hand and the robot hand have similar configurations along the grasp motion.

The synergy space is constructed by grasping a series of objects following the human grasp taxonomy published in [Feix et al., 2016]. In total, 26 grasps are reproducible with the kinematics and structure of the Schunk hand. From the human grasp taxonomy, seven hand configurations can not be performed by the Schunk hand. The grasps are grouped into *power*, *intermediate* and *precision* and all are presented in Fig. 5.4. Additional five open configurations are added to increase the variability of the grasp samples.

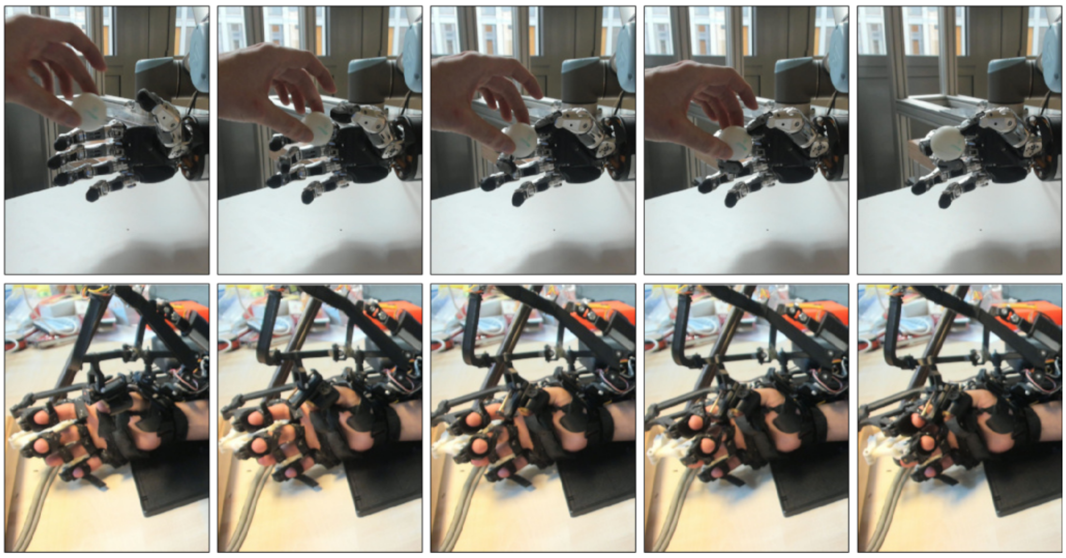

Figure 5.3: A three finger precision grasp is performed by the Schunk hand operated by the U-HEx exoskeleton. Observe the similarities between the hand and the robot configuration during the motion execution.

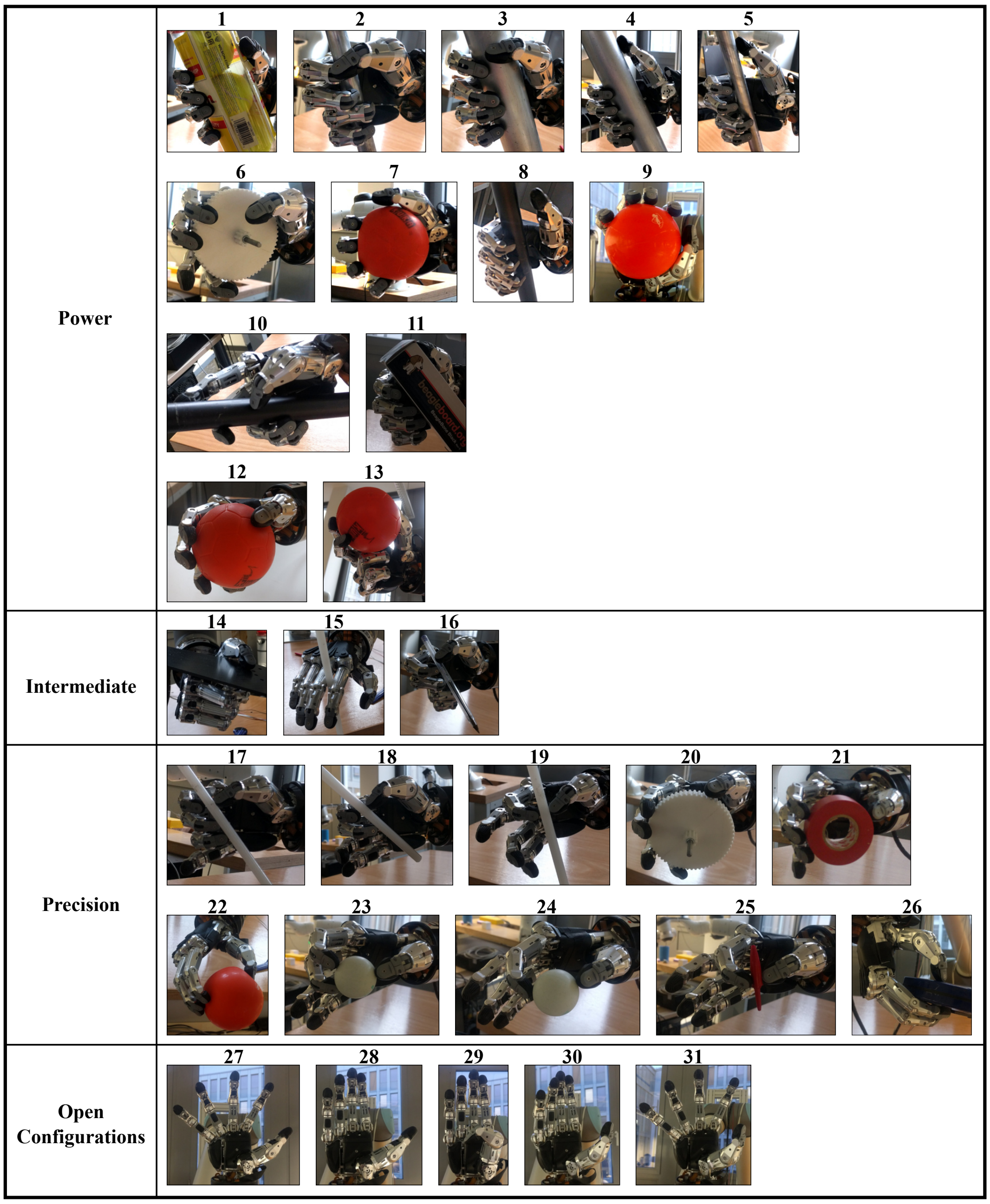

Figure 5.4: Robot grasp configurations used for building the synergy space.

The synergy space is found as follows. The joint position vector $\mathbf{q}$ for each of the n =31 hand configuration are assembled into a matrix $\mathbf{A} = \{\mathbf{q}_1^T - \bar{\mathbf{q}}^T, \dots, \mathbf{q}_n^T - \bar{\mathbf{q}}^T\} \in \mathbb{R}^{n \times q}$, where $\bar{\mathbf{q}}$ is the mean joint position vector. The synergy space is a low-dimensional latent space of the original grasp configurations. This latent space is computed by decomposing the symmetric positive definite matrix $\mathbf{A}^T\mathbf{A} = \mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^T$ into its eigenvectors $\mathbf{Q}$ and its eigenvalues $\mathbf{\Lambda}$, and by building a transformation matrix $\mathbf{L} \in \mathbb{R}^{q \times l}$ with l eigenvectors whose eigenvalues are the highest. The low-dimensional representation $\mathbf{s}$ of a grasp configuration $\mathbf{q}$ is thus formulated as:

$$\mathbf{s} = \mathbf{L}^T(\mathbf{q} - \bar{\mathbf{q}}), \tag{5.1}$$

while the inverse mapping is expressed as:

$$\mathbf{q} = \bar{\mathbf{q}} + \mathbf{L}\mathbf{s}, \tag{5.2}$$

due to the orthonormality of the transformation matrix, $\mathbf{L}^T = \mathbf{L}^{-1}$.

For the synergy space of the Schunk Hand, the explained variance of the synergy space of the Schunk robotic hand with two latent dimensions equals 78% and 88% with three dimensions. These values are similar to the explained variance observed in human grasps based on recordings of 15 joints [Santello et al., 1998]. For three principal components, the explained variance is equal to 90%, and 84% for two components. Compared with the UB hand IV with 15 DOFs, the results are also alike: 75% and 90%, for two and three components, respectively [Ficuciello et al., 2014].

5.2.1 Inverse Kinematics in Synergy Space

The synergy space is built to infer a low dimensional representation of a robot grasp. The inferred configuration, however, can result in self-collision. An iterative constrained inverse kinematics solver is then proposed as an optimization problem to adapt inferred grasp configurations to be collision-free grasps. The solver is expressed directly in the synergy space in order to reduce the dimensionality and correspondingly the runtime of the optimization problem, e.g., to reduce the time of Jacobian computations in higher dimensional joint spaces. A synergy vector $\mathbf{s}$, is then computed by an inverse kinematics solver formulated as:

$$\mathbf{s}_{i+1} = \mathbf{s}_i + \Delta\mathbf{s}_i. \tag{5.3}$$

In each optimization step, a quadratic programming problem is solved expressed as:

$$\begin{aligned} \underset{\dot{\mathbf{s}}}{\text{minimize}} \quad & \frac{1}{2}\dot{\mathbf{s}}^T\mathbf{J}^T\mathbf{J}\dot{\mathbf{s}} + \mathbf{r}^T\mathbf{J}\dot{\mathbf{s}} \\ \text{subject to} \quad & \mathbf{G}_k\dot{\mathbf{s}} \le \mathbf{h}_k, \end{aligned} \tag{5.4}$$

where the Jacobian matrix $\mathbf{J}(\mathbf{s})$ transforms a vector in the task space to the synergy space, $\mathbf{r}$ is the residual or error of the task, $\mathbf{G}_k$ stands for the the Jacobian matrix of the k^{th}

constraint that maps a point in the constraint space to the synergy space, and $\mathbf{h}_k$ represents the residual of the k^{th} constraint. A task defined in synergy space will result in a Jacobian matrix equal to the identity matrix. The optimization problem is solved employing an off-the-shelf quadratic programming solver.

The IK solver is intended to generate collision-free grasps as little deviations as possible from the inferred configuration. Avoiding self-collisions is then part of the constraints of the optimization problem. The Jacobian of this constraint $\mathbf{J}_{self} = \frac{\partial d}{\partial \mathbf{s}}$ is computed numerically, where d describes the penetration distance between two colliding bodies. Collision checking operations are speed up by approximating the geometry of the bodies through primitives (Fig. 4.8). Because the self-collision constraint is formulated as a constraint, and the task, i.e., tracking a target synergy vector, is part of the cost function, the task can be violated in favor of avoiding self-collisions. In this manner, an inferred grasp configuration initially in collision will modify its target task to get rid of the self-collision. The resulting collision-free grasp configuration is finally mapped to the joint space for the motion interpolation and low level commands to the robot hand.

5.3 Learning Postural Synergies

In this section, we described how grasp configurations are learned according to the shape of the objects. In this manner, a robot hand grasp can be inferred only by observing the instances. The objects shapes are represented by their latent shape descriptor using the shape space approach described in Chapter 2. The grasp configurations, are described by a synergy vector as explained in Section 5.2. A model is trained in a supervised manner called the *synergy learner*. Because the inference of synergy vector depends on the shape space and synergy space, they both need to be trained as requisite for training the synergy learner.

The training process of the synergy learner is illustrated in Fig. 5.5. For each object category, a different synergy learner is trained. In addition, the shape space of each category needs to be trained. The synergy space depends on the robotic hand, thus the same synergy space can be employed across multiple object categories. To train a synergy learner of a category, all training object instances are grasped to get a synergy vector for each instance. To speed up this data acquisition time, the exoskeleton is used to operate the robot hand. Moreover, the shape of the training instances is computed by finding their shape descriptor in the shape space. In this manner, each object instance has a shape descriptor associated to a synergy vector (grasp configuration).

Even though a functional grasp imposes several constraints in the manner how to grasp objects, there exists frequently not a unique suitable functional grasp, but a set of them. In other words, a shape descriptor $\mathbf{x}$ can be assigned to different grasp synergy vectors. Based on this observation, the synergy learner is modeled as a distribution of grasps depending on the shape parameters. Specifically, a Gaussian Process (GP) is employed to model

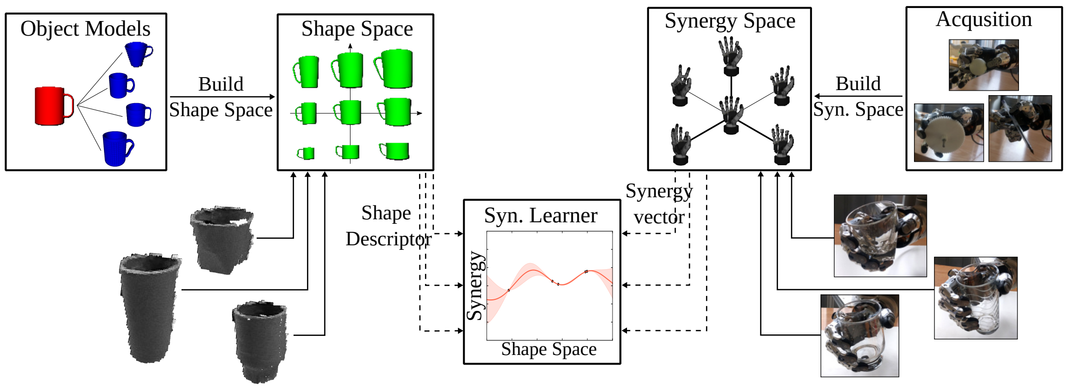

Figure 5.5: Training of the synergy learner. The shape space of the object category and the synergy space of the robot hand are computed a priori. All objects of the training set are grasped to get corresponding synergy vectors. By means of the shape space registration, the shape descriptors of all training instances are found. The synergy learner is then trained by associating the shape descriptors with their respective synergy vectors.

each of the synergy components. In this manner, a shape descriptor $\mathbf{x}$ will be mapped to l Gaussian distributions, where l is the cardinailty of the synergy space. It is therefore assumed that the synergy values are Gaussian distributed for a given shape parameter. All GPs use a Radial Basis Function kernel.

In inference, the shape learner infers a synergy vector given a shape space descriptor computed online as described in Section 2.4. This synergy vector is constructed by combining the inferences (mean predictions) of the individual Gaussian Processes. The inference process is illustrated in Fig. 5.6.

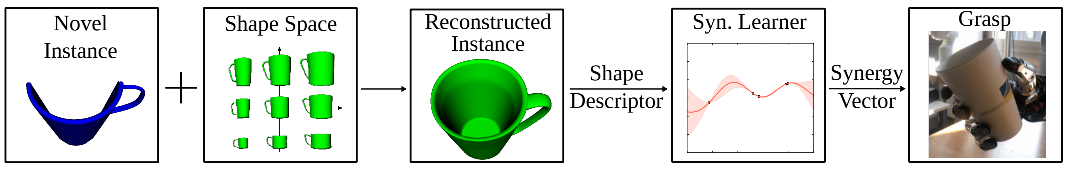

Figure 5.6: Inference of grasp configurations by the synergy learner. The observed instance is registered non-rigidly to obtain a shape descriptor $\mathbf{x}$. As a result the object model is reconstructed. Based on $\mathbf{x}$, a synergy vector is inferred by the synergy learner by combining the mean predictions of the GPs.

5.4 Evaluation

The synergy learner presented in this chapter is tested with the *Sphere* and *Glass* categories. The 3D models for building the shape spaces were obtained from online databases [2,3]. Additional training instances are generated automatically by performing basic transformations on existing instances. Such transformations include scaling in single and combined coordinate axis, and projective transformations. Each category defines its own set of object transformations. Implausible shapes are then removed manually by experts. The shape spaces are built from virtual models only. The object models are modeled as point clouds obtained by ray-casting the 3D meshes and by down-sampling the results by a voxel filter.

The shape spaces are trained with 9 and 16 samples for the Sphere and Glass category, respectively. Interestingly, the shape space of the spheres learns the global scale transformation, which is the only parameter to fully characterize this category. Similarly, the shape space of the glass category learns a representation of the diameter, length and global scale.

The synergy learners of all categories are trained only with real data. Controlling the robot hand with the exoskeleton allows a faster data acquisition compared to simulation. The synergy vectors are calculated by using Eq. (5.1). The objects are perceived by the Kinect v2 RGB-D camera integrating a tabletop segmentation and a voxel filter to reduce the number of points and to speed up subsequent computation time. Because of the transparent surfaces of the glasses, the surfaces are covered by a non refracted material in order to be perceived. The shape descriptor of each training instance is then found by the shape space registration. Finally, the Gaussian Processes are trained associating a shape descriptor with their corresponding synergy vector. Point clouds of Glass training instances with their corresponding grasps are shown in Fig, 5.7.

Initial evaluation of the synergy learner is performed in the Gazebo physics-based simulator for the Glass category. The Schunk hand is attached to a UR10 robot manipulator

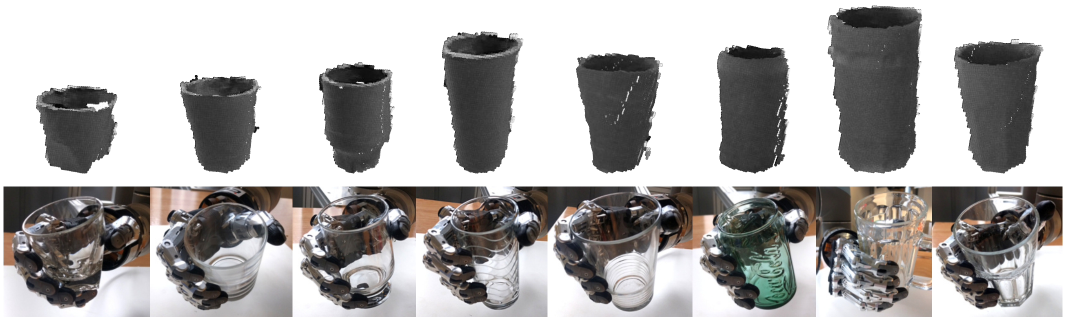
Figure 5.7: Training Glass instances for the synergy learner. Top: observed point clouds, and bottom: the corresponding grasps performed through teleoperation with the exoskeleton.

[2] GrabCad: https://grabcad.com/library

[3] 3DWarehouse: https://3dwarehouse.sketchup.com/

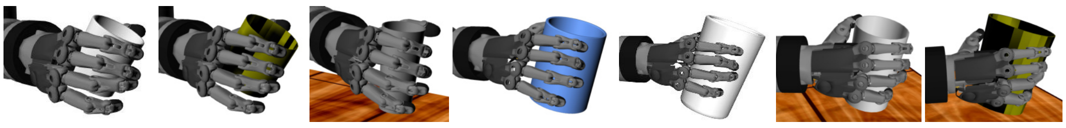
Figure 5.8: Grasp configurations inferred by the synergy learner tested in simulation. All glasses are shown for the first time and are grasped successfully. All objects are held in the air to guarantee that only the robot hand has contact with the glasses.

with 6 DOFs. The objects are placed in a known pose. The approach motions are given such that only the generation of multi-finger grasp configurations can be solely evaluated. Approaches as described in Chapter 4, can be implemented to compute the approaching motions. After the inferred grasp configuration is reached, the robot hand is lifted. If the objects remain in the hand after 10 seconds, the trial is considered as successful. In total, all seven instances are grasped and lifted. Snapshots of the final grasped drills are presented in Fig. 5.8.

Real robot experiments are carried out with the Schunk hand and the UR10 robot manipulator. For the Sphere category, three out of four instances were successfully grasped. The inferred grasp configurations are presented in Fig. 5.9. The Schunk hand is operated through a position controller. In this manner, the quality of the synergy learner for generating grasp synergy vectors can be evaluated. Note that an additional strategy to close the hand until certain current values are reached will allow to grasp the failed instance.

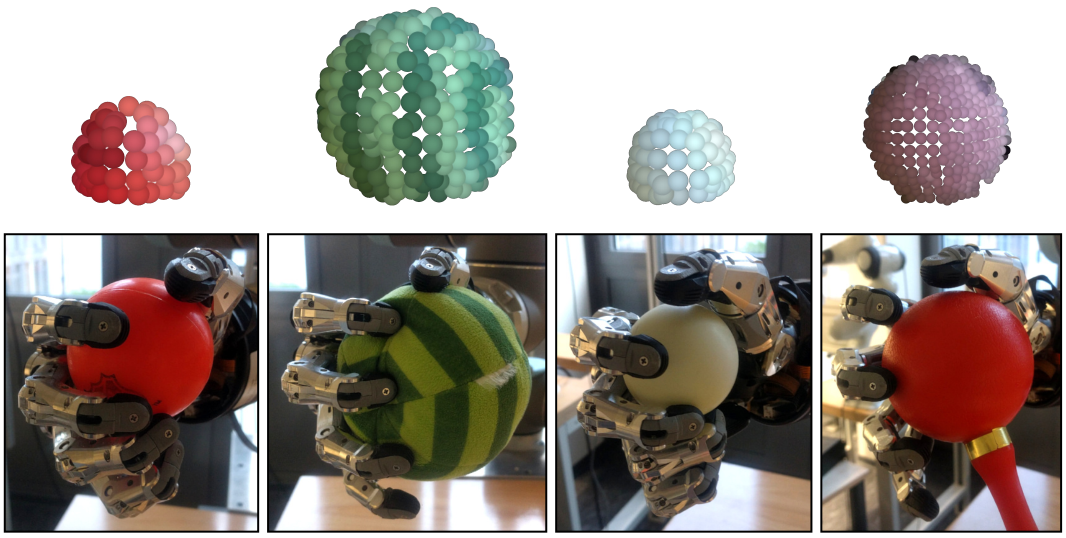
Figure 5.9: Synergy learner evaluation on real spheres. The first three objects from left to right are successfully grasped. The last grasp failed whose inferred grasp is shown for comparison.

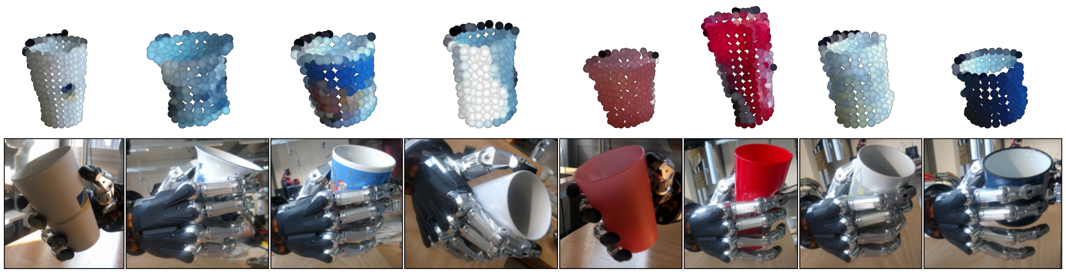

Figure 5.10: Synergy learner evaluation on real glasses. All the instances are novel to our approach and are successfully grasped.

The input to our approach is a point cloud that is a partial observation given by a single view. This demonstrates the applicability of our approach for online grasping.

Finally, our approach is evaluated with real instances of the Glass category. The input point clouds and the resulting grasp configuration of the testing instances are shown in Fig. 5.10. All eight objects are successfully grasped with the inferred configurations given by the synergy learner. The inference time is in average 10 ± 0.8 seconds, which demonstrates again that this approach can be implemented for online applications.

5.5 Discussion

This chapter has presented an approach to infer grasp configurations according to the shape of the objects. The high dimensionality of the multi-finger hand is addressed by creating a synergy space of the robot hand. On the other hand, the object shape is described by latent parameters of the shape space of the category. The performance of the approach has been evaluated in simulation and real robot experiments. One of the main advantages of the method is their applicability to online scenarios (planning average time is 10 seconds). This time can be decreased by further code optimization of the shape space registration.

The approach explained in this chapter removes the assumption of the grasp transfer approach made in Chapter 4 about the fixed motions of the robot hand. Both approaches combined, the grasp transfer and the synergy learner, allow to control arm manipulators and robot hands to perform autonomous grasping operations. In order to capture more variability of grasp configurations in a latent space, non-linear subspace methods such as GP-LVM seem to be a promising alternative [Romero et al., 2013]. In addition, the overload of establishing grasp configurations for training the synergy learner can be reduced by autonomous grasp generators. For this, the implementation of reinforcement learning methods seems to be an interesting option.

Part II

Walking

6 Bayesian Optimization of Bipedal Gait Stabilization

"We do not learn from experience...
we learn from reflecting on experience."
— John Dewey

In Part I, several approaches have been presented that address the robot grasping problem. The joints of the robot arm and the robot hand have been controlled to provide autonomous grasping capabilities to robotic platforms. In this second Part of the thesis, the emphasis is put on the whole body to address the walking problem of humanoid robots. Initially, an analytical gait based on a Central Pattern Generator (CPG) is optimized to parameterize higher level balancing actions. The resulting optimized values lead to a faster and more stable gait. Later, the analytical gait is completely removed and a learned locomotion controller is developed that provides omnidirectional capabilities to biped robots. This gait is learned without providing extra information to the robot such as CPGs or dynamical models. The robot dynamics is captured by the learned model trained by means of Deep Reinforcement Learning (DRL) methods. The optimization approach is presented in this chapter, while the learned locomotion controller is explained in Chapter 7.

Analytical walking controllers often require the parameterization of high level modules that affect the performance of the gait. Although these parameters can be set manually, there is no guarantee of optimality. Optimization methods are then a natural alternative to find the value of these parameters. However, the cost associated to the optimization, including time and wear off of the system, might make this alternative impractical. Even though the optimization can be carried out in simulation, that is cheap and does not wear off the robotic platform, the results are not easily transferable because of the reality gap, i.e., the difference between the simulated and the real robot. On the other hand, performing the optimization directly with the real robot is time consuming and might incur in hardware damages.

In this chapter, a gait optimization method is proposed that allows to combine simulation and real robot experiments. In this manner, sim-to-real transfer is not required and the system wear-off is reduced. This combination happens inside a Bayesian optimization

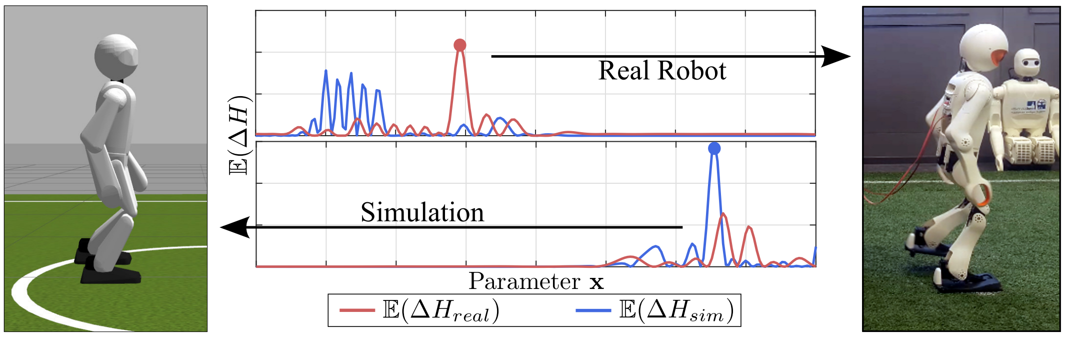

Figure 6.1: Gait optimization by collecting experiences in simulation and with the real robot. Multiple sources of information are combined to learn gait parameters $\mathbf{x}$. The largest relative entropy $\mathbb{E}(\Delta H)$ of a cost function J determines where the next sequence is performed: in simulation or with the real robot. By introducing experiences gathered in simulation, the number of walking sequences performed with the real robot, and consequently its wear off, is reduced.

framework that selects the next candidate to evaluate based on the point that provides more information. This point can represent an evaluation of a set of parameters in simulation or on the real robot. The most informative point is defined through the relative entropy of the cost function to optimize. Fig. 6.1 illustrates this process. A real robot experiment is carried out when the point with the highest relative entropy is higher than the relative entropy of all points in simulation, and vice versa.

Several state-of-the-art walking controllers require fine tuning that is performed by experts due to the complexity of the controllers. This imposes a limitation on the use of these approaches: if experts are not present, the gait can not be modified. In recent years, novel learning approaches has been presented to reduce the amount of expert knowledge for tuning locomotion controllers [Calandra et al., 2014; Rai et al., 2018]. The parameter estimation happens as the results of a series of independent experiments. In this experimentation process, the sample-efficiency of the learning approaches plays a key role to reduce the time and system wear off when performing real robot experiments. In the approach presented in this chapter, the amount of real experiments is reduced by incorporating experiences gathered in simulation and by a sample-efficient learning method based on the relative entropy of the cost function. In addition, if expert knowledge is available, this can be used to guide the learning process, e.g., by defining initial configurations and parameter bounds.

The approach proposed in this chapter is evaluated on the igus Humanoid Open Platform [Allgeuer et al., 2016]. Its gait is based on an open loop pattern generator. In order to be reactive and more robust against perturbations coming, e.g., from ground surfaces or from external pushes, the gait possesses a series of basic mechanisms such as movement of the arms or Center Of Mass (COM) placement. The sensitivity or activation values of these mechanisms constitute the parameters to optimize.

Gait optimizations, as the proposed in this chapter, play a key role especially in robot competitions such as the humanoid league [1] at RoboCup [2]. Robots that move faster or exhibit a more stable and reliable gait have a clear advantage over their opponents. The approach presented here was in part motivated by the participation of the NimbRo team [3] in the RoboCup's humanoid league.

The approach described in this chapter has been published in [Rodriguez et al., 2018b]. The accompanying online Video 6.1 [4] gives an overview of the approach presented in this chapter and shows the igus humanoid robot traversing a rough terrain using an optimized gait as result of applying the method proposed here.

6.1 Related Work

Several optimization methods proposed so far for flying and humanoid robots rely on Bayesian learning because of its high sample efficiency [Akrour et al., 2017; Calandra et al., 2014; Deisenroth et al., 2015; Heijmink et al., 2017; Rai et al., 2018; Röfer, 2005]. Berkenkamp et al. [2016], for example, formulated the tuning process of quadrotor controllers as a Bayesian optimization problem. Similarly, Marco et al. [2016] integrated Bayesian optimization with optimal control to estimate parameters of LQR regulators. Deisenroth et al. [2015] followed a similar Bayesian optimization method to parameterize controllers of a cart-pole system. A different approach was taken by Akrour et al. [2017] who proposed a search distribution to carry out a direct optimization process, however, since the optimization is performed locally, its expressivity is lost on a global scope.

Learning approaches have also been applied for locomotion [Calandra et al., 2014; Heijmink et al., 2017; Hengst et al., 2011]. Calandra et al. [2014] followed an imitation learning approach and learned a stable gait with a real biped robot given a target trajectory. That approach only performed real robot experiments using Bayesian optimization to select candidates to evaluate. A significant wear-off of the robotic system is nevertheless expected because of the high number of executed experiments (walking sequences). In addition, Heijmink et al. [2017] attempted to learn impedance profiles and gait variables for a quadruped robot in simulation. This was achieved by minimizing a cost function containing speed tracking, energy consumption, joint limits and torques by means of the PI^2 algorithm in a Reinforce Learning (RL) fashion. Following a RL approach, Hengst et al. [2011] developed an approach that infers the stance leg and ankle joint position of

[1] https://humanoid.robocup.org/

[2] https://www.robocup.org/

[3] http://nimbro.net/Humanoid/index.html

[4] Video 6.1: https://zenodo.org/record/3991986/files/chapter_6.mp4

the swing foot of a humanoid robot. Interestingly, that method has been evaluated with a real robot.

Successful examples of simulation to real transfer have been presented before. Farchy et al. [2013], for instance, estimate simulator properties such that walking sequences performed in simulation and with the real robot are similar based on a defined metric. Hanna and Stone [2017] extended this approach by inferring the dynamics of the simulator by providing error signals between taken actions in simulation and the real world. They accomplished an increment of 43% in the maximum walking velocity of a NAO robot compared with the latest analytical gait at the time their work was published. Recently, Rai et al. [2018] have built a low-dimensional space in simulation to transfer gait parameters on a real Atrias robot. That approach effectively uses simulation data and even though a large amount of data needs to be precomputed, only 20 iterations were required to execute the gait on the real robot.

6.2 Background

6.2.1 Bayesian Optimization

Bayesian optimization aims to find a global optimum $\mathbf{x}^*$ of a cost function $f(\mathbf{x})$ by means of statistical models. The optimization is gradient-free and characterized to be sample-efficient. This property is especially relevant when the evaluation of the optimizing function is expensive. For instance, performing a walking sequence with a real platform which implies wearing off the system and consumes time. The optimum is the result of the minimization of a posterior mean function. Typically, Gaussian Processes (GP)s are employed to model the underlying statistical model of the Bayesian optimization.

The goal of a Gaussian Process is to build a non-linear mapping between an input and a target manifold. This mapping is the result of fitting a statistical model using a training dataset $\mathcal{D} = \{(\mathbf{x}_i, y_i)|i = 1, \ldots, n)\}$ with n observations, and some *prior* knowledge. The observations are represented with additional white Gaussian noise $\epsilon \sim \mathcal{N}(0, \sigma^2)$ in the form:

$$y = f(\mathbf{x}) + \epsilon \tag{6.1}$$

$$f(\mathbf{x}) \sim \mathcal{GP}(\mu(\mathbf{x}), k(\mathbf{x}_i, \mathbf{x}_j))\,, \tag{6.2}$$

where $\mu(\mathbf{x})$ represents the prior mean, and $k(\mathbf{x}_i, \mathbf{x}_j)$ is the covariance or kernel function. The prior mean can be as simple as a uniform distribution.

The non-linearity of the mapping is attributed to the kernel function that transforms the input space into a higher-dimensional feature space. One of the advantages of using kernels is that the shape of the feature space (which might be very complex) does not need to be known in beforehand, only the inner product of the input vectors needs to be performed. The kernel function $k(\mathbf{x}_i, \mathbf{x}_j)$ describes the similarity between the posteriors

$f(\mathbf{x_i})$ and $f(\mathbf{x_j})$ of two input vectors $\mathbf{x}_i$ and $\mathbf{x}_j$. A low value of $k(\mathbf{x}_i, \mathbf{x}_j)$ tells that the posterior $f(\mathbf{x}_j)$ has a poor influence on $f(\mathbf{x}_i)$, and vice versa.

In Bayesian optimization, an *acquisition function* is in charge of picking the next point $\mathbf{x}$ to evaluate $f(\mathbf{x})$. This acquisition function also manages the trade-off between exploration and exploitation, such that promising points for the optimum are evaluated and the uncertainty about $f(\mathbf{x})$ is reduced.

The *Entropy Search* (ES) method proposed by Hennig and Schuler [2012] is an interesting example of those acquisition functions. According to the expected change of entropy $\mathbb{E}[\Delta H(\mathbf{x})]$, ES selects the next point to evaluate. In other words, ES picks the most informative point defined as the point with the highest entropy change. The acquisition function for the entropy search is expressed as:

$$\mathbf{x}_{t+1} = \arg\max_{\mathbf{x}\in X}(\mathbb{E}[\Delta H(\mathbf{x})])\,. \tag{6.3}$$

The optimum location is approximated by a non-uniform grid, which, upon convergence, will be around the actual minimum. Computational approximations of the ES are described in [Hennig and Schuler, 2012].

6.2.2 Multi-Fidelity Entropy Search

The ES algorithm has been extended by Marco et al. [2017] in order to consider different data sources. This new approach is referred as *Multi-Fidelity Entropy Search* (MF-ES), which was motivated by the lack of sample-efficient methods that combine real experiments with simulations. Fig. 6.2 illustrates two iterations of this optimization process with synthetic data. In case, two sources of information are considered, e.g., simulation and real robot experiments, the cost function defined by the MF-ES has the form:

$$J_{real}(\mathbf{x}) = J_{sim}(\mathbf{x}) + \epsilon_{sim}(\mathbf{x}). \tag{6.4}$$

The MF-ES approach is based on the idea of modeling the cost on the real system J_{real} as the cost in simulation $J_{sim}(\mathbf{x})$ together with a systematic error $\epsilon_{sim}(\mathbf{x})$. The role of the Bayesian optimization is, among others, to learn the non-linear transformation that describes the error ϵ_{sim}. MF-ES models the cost on simulation and its dissimilarity to the physical system by means of two kernel functions, k_{sim} and k_ϵ, respectively. Thus, in overall, the kernel is formulated as:

$$k(\mathbf{a}_i, \mathbf{a}_j) = k_{sim}(\mathbf{x}_i, \mathbf{x}_j) + k_\delta(\delta_i, \delta_j)k_\epsilon(\mathbf{x}_i, \mathbf{x}_j)\,, \tag{6.5}$$

where $\mathbf{a} = (\delta, \mathbf{x})$ is an augmented vector which includes an indicator δ to differentiate between real ($\delta = 1$) and simulated ($\delta = 0$) experiments, $k_\delta(\delta_i, \delta_j) = \delta_i\delta_j$ is a kernel indicator that becomes one when δ_i and δ_j represent both physical experiments. Conse-

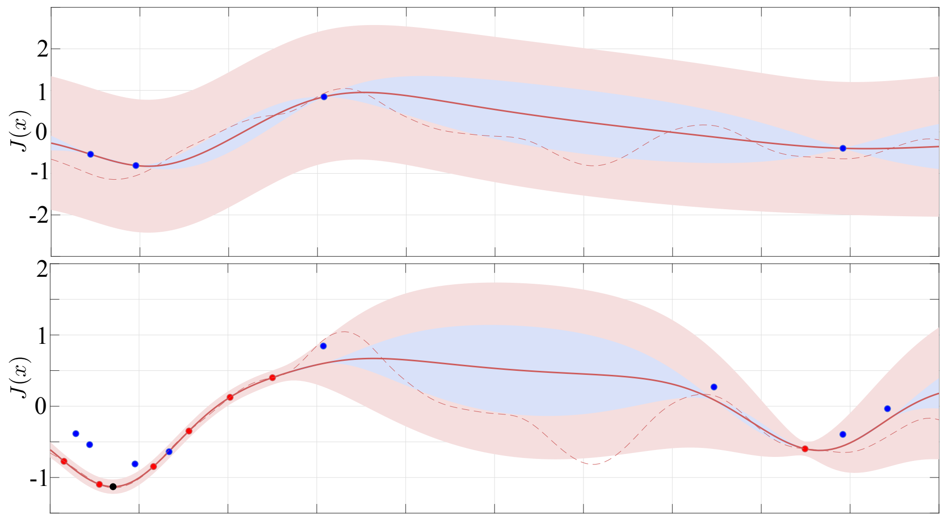

Figure 6.2: Combining real (red dots) and simulated (blue dots) experiments with synthetic data. The ground truth cost function is plotted with a dashed red line. Top: initial evaluations in simulation influence the posterior mean of the cost function of the real experiments J_{real}. The posterior variance (blue shaded) is strongly conditioned on the simulation experiments, but the posterior variance (red shaded) of the real experiments is weakly conditioned. Bottom: by performing real-robot experiments, the posterior variance of J_{real} is reduced. The optimum, at that iteration, is displayed with a black dot. The dissimilarity between the simulation and real data is modeled by ϵ_{sim} in Eq. (6.4).

quently, for the posterior, the influence between two physical experiments is expected to be higher compared to two simulated or a simulated and a physical experiment.

The complex non-linear mapping between simulation J_{sim} and real robot experiments J_{real} does not need to be explicitly addressed, because k_ϵ is part of the Gaussian process. Instead, prior information, expressed by a mean and a covariance function, needs to be given. Furthermore, the acquisition function has to be adapted to the multiple data sources, otherwise only physical experiments would be performed since their information gain is higher compared to simulation. δ is integrated into the acquisition function by inserting weight parameters w_i for each of the information sources. Therefore, the acquisition function of the MF-ES is defined as

$$\mathbf{x_{t+1}} = \underset{\mathbf{x}\in\mathbb{R}^d, i\in\{\text{sim,real}\}}{\arg\max} \left(\frac{\Delta H_t(\mathbf{x})}{w_i}\right). \tag{6.6}$$

6.2.3 Bipedal Gait with Feedback Mechanisms

In this subsection, the gait to optimize is explained [Allgeuer and Behnke, 2016]. The bipedal gait relies on a open-loop central pattern generator complemented with feedback mechanisms that counteract external disturbances coming from the terrain or from pushes. The open-loop gait extended the approach presented in [Missura and Behnke, 2013] which was inspired by the work published in [Behnke, 2006]. This open-loop gait initially generates trajectories using a low-dimensional representation called the leg abstract space. This space is an abstraction that allows for an intuitive control aimed for walking tasks compared to joint or Cartesian representations. The generated trajectories are parameterized by a three-dimensional commanded target velocity vector $v = [v_x, v_y, \omega_z]$. From a halt pose of the bipedal robot defined in the abstract space, the gait integrates a set of motions primitives including leg lifting and swinging defined in the same space. Additional motion primitives are incorporated in the inverse space after the resulting abstract pose is converted into the Cartesian space. The resulting pose is finally transformed into the joint space where the final trajectory is commanded to the low-level controllers.

The reactive actions defined by the feedback mechanisms operate according to the orientation of the robot, which is described by fused angles [Allgeuer and Behnke, 2018] (an alternative representation for Euler angles). The sensibility or activation value $\mathbf{u}$ of these feedback mechanisms depend on deviations from desired target values, e.g., from an upright standing pose. These deviations are composed by d_α and d_β that define error values of the fused roll α_B and fused pitch β_B angles, respectively. The components of $\mathbf{u}$ can be interpreted as the strength or magnitude of the respective corrective actions or feedback mechanisms. The following corrective actions are part of the gait: *arm swinging*, *hip movement* (roll and pitch), *COM shifting*, *ankle tilting*, and *support foot tilting*. The introduction of these mechanisms ultimately modifies the trajectories in the abstract or Cartesian space. From the fused deviations d_α and d_β, a series of filters (mean, derivatives, integrators, and smooth deadband) are applied to produce a PID vector $\mathbf{e} \in \mathbb{R}^6$. In order to obtain the final activation vector $\mathbf{u}$, the PID vector $\mathbf{e}$ is multiplied by a gain matrix $\mathbf{K_a} \in \mathbb{R}^{m \times 6}$, where m represents the number of enabled feedback mechanisms.

6.3 Gait Parameter Learning

In this section, the gait optimization problem is formally defined. Specifically, the activation weights $\mathbf{K}_a$ of the feedback mechanisms are learned based on the Multi-fidelity Entropy Search approach introduced in Section. 6.2.2.

In this approach, the walking stability of the robot is determined by means of the roll and pitch fused deviation angles $d_\alpha = \alpha_{des} - \alpha$ and $d_\beta = \beta_{des} - \beta$. These deviations give an intuitive tilt estimate of the humanoid robot. As stated in Sec. 6.2.3, however, the direct measurements of these deviations contain a large amount of noise. A mean and smooth deadband filters are applied to these deviations values to obtain a proportional part, $e_{P\alpha}$ and $e_{P\beta}$, of the fused feedback vector $\mathbf{e}$. For an entire walking trajectory, a stability walking criterion is defined by integrating the sum of the filtered fused deviation angles, $e_{P\alpha}$ and $e_{P\alpha}$, along the walking sequence duration T:

$$\int_0^T \|e_{P\alpha}(\mathbf{x})\|_1 + \|e_{P\beta}(\mathbf{x})\|_1 dt\,. \tag{6.7}$$

Establishing this stability criterion encourages the robot to avoid falling. In order to speed up the optimization process, additional prior knowledge is provided in form of maximum limits (upper bound) of the optimizing parameters $\mathbf{x}$. This upper bound is integrated as a regularization or penalty term into the cost function together with the stability criterion. This regularization term is inspired by the logistic function and it is expressed as:

$$\nu(\mathbf{x}) = \frac{s}{1 + \exp(-\gamma(\|\mathbf{x}\|_2 - \|\mathbf{x}_{max}\|_2))}\,, \tag{6.8}$$

where $\mathbf{x}_{max}$ is the upper bound of $\mathbf{x}$, s is the strength of the penalization, and $\gamma \in \mathbb{R}$ determines the smoothness of the transition.

The feedback mechanisms make a clear distinction between the sagittal and the lateral direction. This observation is exploited and two sub-cost functions are proposed, one for the lateral direction $J_\alpha(\mathbf{x}_l)$ and one for the sagittal direction $J_\beta(\mathbf{x}_s)$. The parameter $\mathbf{x}$ is correspondingly split in $\mathbf{x}_l$ (lateral) and $\mathbf{x}_s$ (sagittal), according to the directions of motion that the parameters contribute to. $J_\alpha(\mathbf{x}_l)$ and $J_\beta(\mathbf{x}_s)$ are defined as:

$$J_\alpha(\mathbf{x}_l) = \int_0^T \|e_{P\alpha}(\mathbf{x}_l)\|_1 dt + \nu(\mathbf{x}_l), \tag{6.9}$$

and

$$J_\beta(\mathbf{x}_s) = \int_0^T \|e_{P\beta}(\mathbf{x}_s)\|_1 dt + \nu(\mathbf{x}_s)\,. \tag{6.10}$$

The walking sequences in simulation are performed in the Gazebo simulator. The intrinsic noise of the simulation is counteracted by performing a walking sequence with identical parameters several times. In this manner, the cost function in simulation is expressed by the mean of N walking sequences with identical parameters:

$$\bar{J}_{sim,i}(\mathbf{x}) = \frac{1}{N}\left(\sum_{j=1}^{N} \int_0^T \|e_{Pi,j}(\mathbf{x})\|_1 dt\right) + \nu(\mathbf{x}), i \in \{\alpha, \beta\}. \tag{6.11}$$

With the physical system, only one walking sequence is performed per evaluation, i.e., for J_{real} $N = 1$. An additional penalization term is added to the cost function when the robot falls during a walking sequence.

The Gaussian process of the Bayesian optimization is parameterized with a Rational Quadratic (RQ) kernel. Both kernel functions, i.e., k_{sim} and k_ϵ of Eq. (6.5), employ the same kind of kernel:

$$k_{RQ}(\mathbf{x}_i, \mathbf{x}_j) = \sigma_k^2 \left(1 + \frac{(\mathbf{x}_i - \mathbf{x}_j)^2}{2\rho \boldsymbol{l}^2}\right)^{-\rho}, \tag{6.12}$$

with hyperparameters σ_k^2, ρ and $\boldsymbol{l}$. The scalar σ_k^2 corresponds with the problem-specific signal variance, $\boldsymbol{l}$ fixes an influence distance between two points, and ρ is a relative gain for large-scale and small-scale variations.

Often, optimization algorithms are terminated according to a maximum number of iterations. Although this simple condition works fine with problems whose evaluation is fast and cheap to carry out, this condition might lead to unnecessary evaluations and corresponding system wear off. Inspired by the acquisition function of the Bayesian optimization used in this work, a different termination condition is proposed based on the relative entropy $\mathbb{E}[\Delta H(\mathbf{x}_t)]$. In this manner, the optimization is stopped after the relative entropy is below a certain threshold. To reduce the impact of outliers, a filter is applied to the relative entropy values. Without filtering the signal, the optimization might terminate prematurely. This problem is illustrated in Fig. 6.3, with no filtering, the optimization could stop in iteration 80, missing 280 additional iterations. The filter is defined as:

$$(1 - v)\mathbb{E}[\Delta H(\mathbf{x}_{t-1})] + v\mathbb{E}[\Delta H(\mathbf{x}_t)]. \tag{6.13}$$

where $\in [0, 1]$ is a velocity factor. After the first iteration of the optimization, the relative entropy can reach very low values. Therefore, this termination condition incorporates a minimum number of iterations to be enabled. Such low relative entropy values are obtained when a non appropriate prior mean is provided. In addition, a maximum number of iterations guarantees the termination of the optimization in case the minimum value of the relative entropy is not reached. This might happen when the given prior does not model reality well enough.

6.4 Evaluation

Experimental Setup

The approach presented in this chapter is evaluated with the igus Humanoid Open Platform [Allgeuer et al., 2016]. The platform has six DOFs per leg, and three DOFs in each arm. Two additional joints control the head allowing pan and tilt motions. All together, the robot possesses 20 DOFs. The humanoid weights 6.6 kg and is 92 cm tall. All body links are 3D printed and open sourced [5]. Each joint is actuated by Robotis Dynamixel MX

[5] https://github.com/igusGmbH/HumanoidOpenPlatform

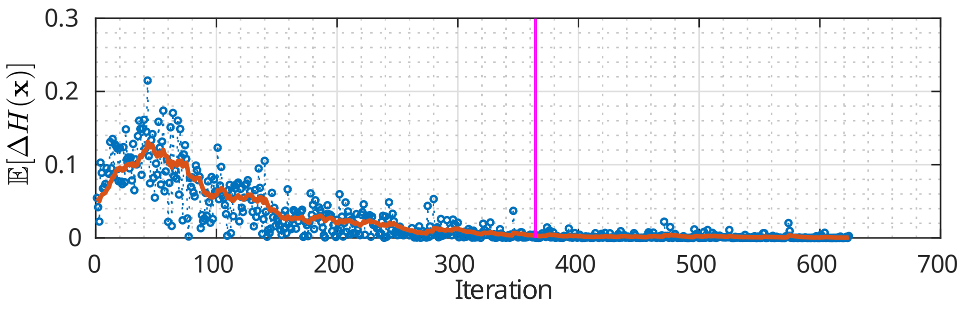

Figure 6.3: The filtered relative entropy (red line) is used as termination criterion of the optimization. The filter defined in Eq. (6.13) is parameterized with $v = 0.9$. Because of the low information gain of the experiments after the vertical magenta line, these experiments can be skipped. Note that a naive condition that terminates the optimization without the filter would terminate prematurely in iteration 80.

servomotors. The actuators are interfaced through a CM730 microcontroller board. An Inertial Measurement Unit (IMU) composed of three axis gyroscope, magnetometer and accelerometer are integrated into this board. An Intel Core i7-5500U running a 64-bit Ubuntu operating system is onboard.

The simulations are carried out in a desktop PC with an Intel Core i7-4890K CPU and 8 GB of RAM. The optimization is also performed with this computer such that the robot PC is used only for performing walking sequences. The body link geometries are approximated trough convex hulls. The simulations are implemented in Gazebo with the Open Dynamics Engine. The simulation is restarted before a new walking sequence is performed. The optimization was implemented in Matlab and the gait in C++. The ROS middleware manages the interprocess communication.

A walking sequence starts with the robot walking on the spot for three seconds, this avoids artifacts coming from the transition of the static pose. Then, the robot is commanded to walk forward at 0.3 m/s. The optimization is applied to learn two different sets of parameters. Firstly, the proportional (P) and the derivative (D) components of the *ankle tilt* corrective action is learned, which is referred here as 2D optimization. And secondly, the PD gains are learned for the *ankle tilt* and the *arm swing* actions which is called here 4D optimization. When performing the evaluation, the other feedback mechanisms are disabled.

The radial quadratic kernels (Eq. (6.12)) are parameterized with $\boldsymbol{l} = \left(\frac{\mathbf{x}_{max}}{8}\right)$ and $\rho = 0.25$. This results in a reasonable exploration and exploitation trade-off. k_{sim} and k_{ϵ} share the same values for $\boldsymbol{l}$ and α. For the 2D optimization, the standard deviations are set to $\sigma_{sim} = 2.48$ and $\sigma_{\epsilon} = 2.07$ for k_{sim} and k_{ϵ}, respectively. On the other hand, the 4D optimization, uses $\sigma_{sim} = 2.07$ and $\sigma_{\epsilon} = 1.79$. In addition, the regularization function $\nu(\mathbf{x})$ (Eq. (6.8)) takes the following values $s = 7.5$ and $\gamma = 6$. This discourages larger parameter values. The values of the prior means, $\mu_{sim} = 53.3502$ and $\mu_{\epsilon} = -37.1385$, are set from initial experiments. Finally, physical experiments are configured to be five times more expensive than a simulated one. Correspondingly, w_{sim} and w_{sim} of Eq. (6.6) are set to 10 and 50, respectively.

Experimental Results

The optimized values are compared with by-experts manually tuned parameters that lead team NimbRo to win the TeenSize of the humanoid league at Robocup 2017 [Rodriguez et al., 2018d].

Initially, the 2D optimization is performed with a total of 126 iterations including 20 walking sequences with the real robot. It means, that for each real experiment five walking sequences were performed in simulation. This highlights the relevance of combining different sources of information in the optimization in order to reduce the costs associated with performing experiments directly on the real platform. The average cost of 15 walking sequences using the manually tuned parameters results in a cost of 13.77, while the sequences with optimized values yield an average cost equal to 9.3. A total improvement of 32% is then achieved by employing this approach. The posterior mean and covariance of the optimization in different iterations are displayed in Fig. 6.4. In addition, the posterior mean considering only real robot experiments is shown at the rightmost plot of Fig. 6.4 in order to stress the contribution of the simulation to the optimization. Note especially, the regions with high cost in simulations, $(x_1, x_2) = (1.0, 0.0)$ and $(x_1, x_2) = (0.0, 0.0)$, where real robot experiments are avoided, represented by the lack of red points (walking sequence with the physical system).

As a baseline, a random grid search is implemented in which parameters are selected by adding noise to the current best guess. The parameters are selected in a greedy manner. A maximum of 25 real walking sequences are carried out and one of them resulted in a robot falling. The average cost of the parameters given by the random search equals 11.04. Although this results are better compared to manually tuned parameters, our approach is still better than the grid search by 15% with a lower number of physical experiments.

The 4D optimization of the PD gains of ankle and arm swing corrective actions performed 301 walking sequences in which 271 were in simulation and 30 with the real platform. So, one real robot walking sequence is carried out for every nine simulated experiments. Again, for the comparison with manually tuned parameters, 15 walking sequences are carried out with the real platform. The manually tuned and optimized parameters yield a cost of 16.28 and 10.38, respectively, which result in an improvement of 35%.

The stability of the gait during five complete trajectories are also compared between the manually tuned and the optimized parameters. The fused angle deviation is measured and its magnitude is integrated over the duration of the walking sequences, i.e., $\bar{D}_\alpha = \int_0^T \mathbb{E}[||d_\alpha||]dt$. The results are displayed at the right of Fig. 6.5. After walking on the spot, the deviation of the manually tuned parameters start diverging from the optimized ones,

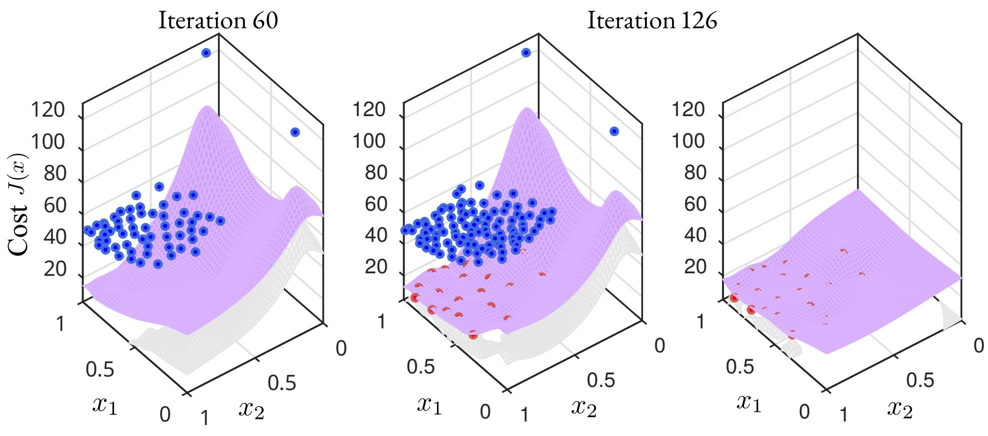

Figure 6.4: Estimated cost function $J(\mathbf{x})$ of the PD activation gains of the ankle tilt by using Multi-Fidelity Entropy-Search Bayesian optimization. The posterior mean and covariance are displayed by the purple and gray meshes, respectively. Initially, the algorithm decided to carry out walking sequences only in simulation (blue points). Only when enough information is collected in simulation, experiments with the physical platform (red points) are performed. The rightmost plot displays the posterior mean composed of only real-robot experiments in order to highlight the contribution of the simulation. Note how the knowledge gathered in simulation shapes the posterior such that real experiments are avoided in some regions.

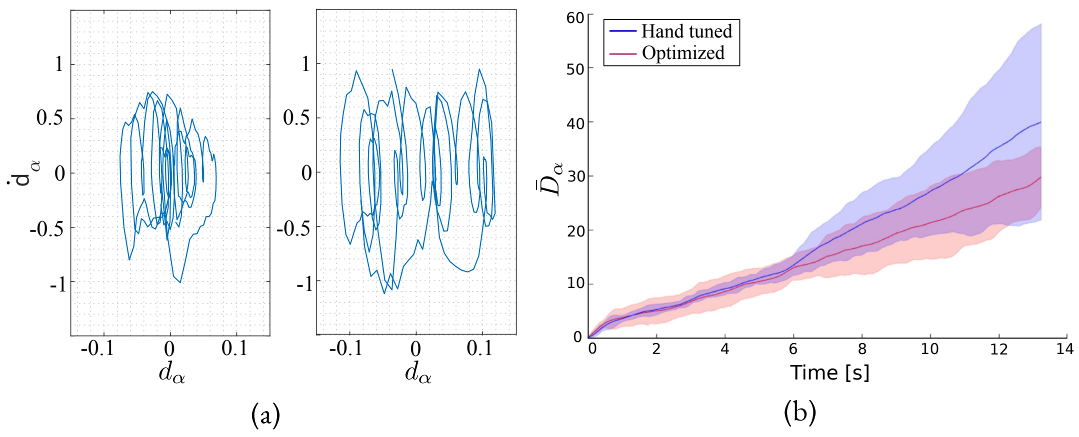

Figure 6.5: a) Phase plots of a walking sequence performed with optimized (left) and manually tuned (right) parameters for the PD gains of the ankle tilt and arm swing corrective actions. For clarity, only one walking sequence is shown but all the executed sequences present a similar behavior. b): integral of the mean absolute fused angle deviation $\bar{D}_\alpha$ of the optimized (red) and the manually tuned (blue) parameters. Values inside two standard deviations ($\pm 2\sigma$) are displayed as shadow regions.

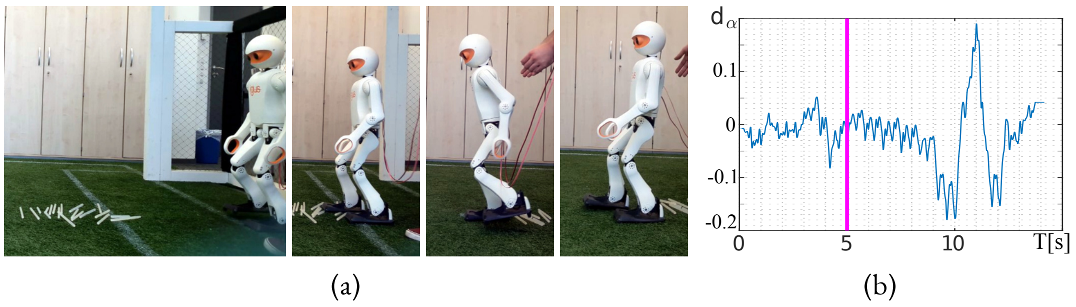

Figure 6.6: Optimized gait evaluated in rough terrain (artificial grass with small debris). a) Initial configuration and snapshots of the robot traversing the rough terrain. b) Fused angle deviation of the robot. The instant when the robot encounters debris for the first time is displayed with a vertical magenta line.

indicating a lower stability. Phase plots of gaits generated with both set of parameters are shown at the left of Fig. 6.5.

Interestingly, during the entire optimization process, the robot does not fall a single time. The experiments in simulation allow to devise regions where parameters result in the robot falling and these values are ruled out implicitly by assigning a high cost. Qualitatively, the humanoid walking with the optimized parameters look more stable and shows in general a more upright torso. Finally, the optimized gait is evaluated against rough terrains, i.e., synthetic grass with small cylindrical debris. Snapshots of the robot traversing this rough terrain are shown in Fig. 6.6.

6.5 Discussion

In this chapter, an approach to optimize high level gait parameters was presented. This method leverages simulation data in order to reduce the number of required experiences with the real robot. In this manner, the hardware wear off is reduced which is especially relevant when operating low-cost robots whose failure ratio is normally higher than more expensive platforms. The optimization employs a Bayesian framework in which the candidates to evaluate are selected based on the relative entropy, or, in other words, where the most gain information is expected. In the evaluation, (Section 6.4), it was shown that experiences gathered in simulation established regions that might cause falls and these zones are discarded to perform walking sequences with the physical platform. This was evidenced by the fact that the real robot did not fall a single time during the optimization. Moreover, the optimized parameters exhibit a faster and more stable gait.

The success of this optimization method depends on the exploration of the optimization space. the larger the space, the more samples are required. This imposes a limitation on the number of parameters to optimize. In addition, the approach presented in this chapter requires an analytical gait to optimize. In the next chapter, the optimization approach and the analytical gaits are replaced by a learned controller.

7 DeepWalk: Omnidirectional Bipedal Gait by Deep Reinforcement Learning

"Success consists of going from failure to failure without loss of enthusiasm."
— Winston Churchill

In the previous chapter, a learning method was proposed that optimizes high level gait parameters in order to increase the stability and speed of a humanoid gait. The optimization method relies on an existing analytical gait based on a Central Pattern Generator (CPG) and stabilizing feedback mechanisms [Allgeuer and Behnke, 2016]. The approach presented in this chapter goes a step further and replaces the analytical gait by a fully learned walking controller. In this manner, the modeling of complex high-dimensional dynamics is avoided. Analytical approaches for robot locomotion normally introduce simplifications of the robot dynamics to meet real time constraints. Instead of estimating parameters of several modules, such as central pattern generators and individual feedback mechanisms, the approach presented in this chapter, requires tuning of only the learning method.

The learned walking controller relies on simulation data and novel Deep Reinforcement Learning (DRL) methods. These kind of approaches are inspired in part by the nature of human learning, which consists of an iterative reward and punishment process. In this approach, the model dynamic is not explicitly modeled but learned and represented through a neural network that accumulates knowledge from previous experiences. As a replacement for analytical gait modules, the approach presented here does not rely on any other locomotion modules, such that the robot starts learning from scratch by exploring and exploiting its action space.

Specifically, a novel fully omnidirectional walking controller for humanoid (biped) robots is presented. With this controller, the robot is able to walk forwards, backwards, laterally, and to turn on the spot. More interestingly, the humanoid can combine these individual motion capabilities, and so, walking diagonally or turning during walking is feasible. Fig. 7.1 shows a humanoid robot exhibiting omnidirectional capabilities, i.e., turning left during forward walking.

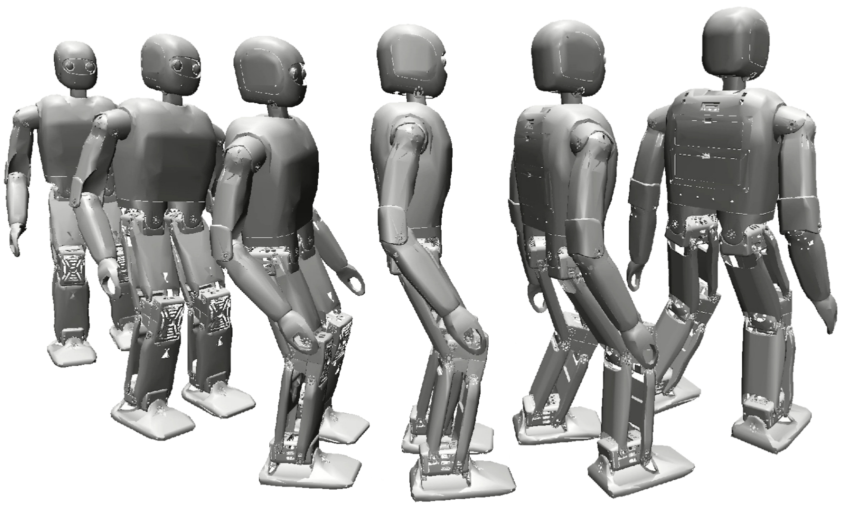

Figure 7.1: Omnidirectional walk learned by Deep Reinforcement Learning. The humanoid turns left during forward walking.

The learned omnidirectional walking capabilities emerge thanks to the incorporation of a novel curriculum strategy that acts as a velocity scheduler. This encourages the agent to learn simple walking capabilities before attempting to learn more complex walking patterns. This curriculum strategy serves then as a guide that enables the acquisition of complex locomotion tasks. Thanks to the curriculum learning, a single policy (neural network) is learned. The parameter share, that happens by using a single neural network, facilitates information transfer and allows to learn speed changes. Moreover, it avoids training single velocity policies separately and the corresponding engineered efforts to combine them. Although previous approaches such as [Yu et al., 2018] and [Hwangbo et al., 2019] also used curriculum learning, in this work, a novel curriculum strategy is presented that enables bipedal robots to walk omnidirectional without reference motions, which has not been demonstrated before.

In contrast with previous approaches that learn walking controllers using DRL methods, our approach does not require reference motions. For this reason, our learning method can be applied to humanoid robots with different kinematics with almost no additional effort. More importantly, analytical gaits or human-to-robot mappings via motion capture systems are not required, which contributes to increasing the applicability of our approach to other platforms. To guide the learning process and obtain anthropomorphic motions, out approach employs a nominal pose that acts as a regularizer.

Our approach incorporates Beta policies instead of the commonly used Gaussian policies. Although Beta policies have been presented for Reinforcement Learning (RL) tasks in general, these policies have not been adopted so far for learning locomotion controllers. In this chapter, we aim to attract the attention of the robotics community of the benefits of using such policies. Beta policies solve the weighting problem of the torque reward term used normally as regularizer. Low values of the torque weight might produce unnatural gaits, while high weights might discourage the agent to move. Moreover, these policies allow to define actuator speed limits elegantly in the learning problem which are normally not explicitly modeled, and thus, it contributes to the generation of more realistic motions.

This chapter is structured as follows. In Section 7.1, related learning methods for bipedal locomotion coming from the animation and robotics communities are presented and compared with our approach. Next, in Section 7.2.1, a brief introduction to the state-of-the-art stochastic model-free deep reinforcement learning methods is given. Our approach relies on these methods. The biped locomotion problem is then formulated and developed in Section 7.3. This includes the definition of the state and action spaces, reward function and curriculum. Later, strategies for transferring the gait to the real robot are presented. Finally, the learned locomotion controller is evaluated in Section 7.5. Especially, the omnidirectional and push-recovery capabilities are evaluated.

The approach described in this chapter has been accepted for publication in the Proceedings of the IEEE International Conference on Robotics and Automation (ICRA) [Rodriguez and Behnke, 2021]. The omnidirectional capabilities of the learned locomotion controller are presented in the accompanying online Video 7.1 [1].

7.1 Related Work

In the recent past, learning approaches have been predominantly applied for gait parameter optimization [Calandra et al., 2014; Farazi et al., 2013; Rai et al., 2018; Rodriguez et al., 2018b], as the approach presented in Chapter 6. These methods, however, depend on specialized and engineered modules such as CPGs, trajectory planners, and robot dynamics models. Often, the high dimensionality and complexity of the robot dynamics render the control problem computationally intractable. To address this issue, learning a complete walking controller, which includes the dynamics, seems to be a promising alternative.

Initial successful efforts to learn locomotion controllers have been previously demonstrated on animation characters on physics-based scenarios [Heess et al., 2017; Peng et al., 2018; Peng et al., 2017; Schulman et al., 2017; Yu et al., 2018]. Heess et al. [2017] exhibit robust locomotion patterns with multiple characters to pass over obstacles in challenging environments. That approach is based on a curriculum strategy that is not applied to the agent itself, but to the environment. The authors demonstrated that complex maneuvers can emerge from simple reward function with minimal craftsmanship. The resulting motions, however, do not exhibit anthropomorphic walking patterns. In addition, the observation space contain exteroceptive ground truth data such as the global position of the agent, and so it is not easily applicable to robotic systems in real scenarios.

[1] Video 7.1: `https://zenodo.org/record/3991986/files/chapter_7.mp4`

In a similar manner, Schulman et al. [2017] applied DRL approaches, namely Proximal Policy Optimization (PPO), on a humanoid model to reach given target poses. Although the walker is able to move in different directions it does not exhibit natural motions. The agent lacks of realistic robotic models which makes the learning task much easier and access to perfect information from the simulator is guaranteed. Moreover, that walker is not able to walk at desired velocities, its goal is only to reach given poses. Our experiments in Section 7.5 show that applying vanilla PPO, as in [Schulman et al., 2017], to a realistic robot model does not produce any walking locomotion controller. This was also demonstrated in [Yu et al., 2018].

From the character animation community, Peng et al. [2017] obtained extraordinary results training walking biped controllers by incorporating motion capture data and target foot-placement poses into a reinforcement learning framework. Later, Peng et al. [2018] proposed DeepMimic which generates highly dynamic motions as cartwheels, backflips and rolls by mimicking motion data. A limitation of this work is the overload to train a different policy for each of the mimicked motions. In addition, [Peng et al., 2017] and [Peng et al., 2018], use simplified models with unrealistic actuation which exceed current robot capabilities.

In a different approach, Yu et al. [2018] was able to produce symmetric locomotion without reference motions. This allows to apply this method on different character morphologies with minimal effort: no kinematic mapping or motion processing modules are needed. The learning process is guided through an assistant that helps the agent to keep the balance. The assistant help is progressively removed setting a curriculum learning strategy. Forward and backward motions can be achieved with this method but omnidirectional capabilities are not present.

In the robotics community, few recent works have implemented deep reinforcement learning approaches for the locomotion problem [Hwangbo et al., 2019; Tsounis et al., 2019; Xie et al., 2018; Xie et al., 2019; Yang et al., 2018]. Hwangbo et al. [2019] proposed a velocity-conditioned policy which was successfully transferred to a real Anymal robot. The key idea for the transfer was the representation of the actuation dynamics by means of a neural network. Thus, the fidelity of the simulation is increased which allows for one-shot sim-to-real transfer. The gait was developed for quadruped robots, whose balance problem is less demanding compared to bipedal gaits.

Yang et al. [2018] presented one of the first attempts to implement DRL approaches on humanoid robots. Push-recovery capabilities were learned in simulation for the Valkyrie robot by incorporating capture step equations into the reward function of a RL method. Only the lower body joints were commanded together with a torso pitch joint. Moreover, locomotion capabilities such as forward walking were not learned, only reactive motions against external pushes were inferred.

Xie et al. [2018] trained a walk controller to produce a gait in the sagittal direction with the biped robot Cassie. The policy is similar to [Peng et al., 2018] and thus it requires reference motions to track. Different walking velocities require different reference motions. Consequently, separate control policies need to be trained and interpolated to produce transitions between commanded velocities. This approach was extended in Xie et al. [2019] to transfer the policies to the physical platform. That policy does not achieve omnidirectional walking. The gait lacks the combination of motions in several directions: walking diagonal or walking forward during turning is not possible. Moreover, separate policies are trained and later combined in a single policy. Our approach, on the other hand, allows to train a single policy directly capable of walking in several directions simultaneously. This facilitates speed changes as showed in our experiments and additionally avoids engineered solutions to combine separate policies.

7.2 Background

7.2.1 Stochastic Deep Reinforcement Learning

In a reinforcement learning problem, experiences with the environment are accumulated to train a policy π that allows an agent to solve a given task. To model this task, a Markov Decision Process (MDP) defined by the tuple $\{S, \mathcal{A}, P, \gamma, r\}$ is employed, where $S \in \mathbb{R}^n$ defines the state space, $\mathcal{A} \in \mathbb{R}^m$ represents the action space, $P : S \times \mathcal{A} \mapsto S$ models the dynamics of the system, $\gamma \in [0, 1]$ is a discounted factor, and $r : S \times \mathcal{A} \mapsto \mathbb{R}$ is a function that rewards or punishes an action a_t taken in state s_t after interacting with the environment at time step t.

Reinforcement learning approaches can be divided into model-based and model-free. The former leverages on the model P to find an optimal policy, while the latter construct directly a policy without exploiting or modeling explicitly the dynamics. The walking controller presented in this chapter constructs a parameterized policy π_θ directly by maximizing a cost function $J(\theta)$ with respect to the parameters θ, rendering a model-free reinforcement learning problem. In contrast with discrete action spaces, in continuous control problems, a stochastic policy $\pi_\theta(a|s)$ is formulated as a probability distribution of taking an action a_t given a state s_t. In other words, instead of defining a single action for a state s_t, a probability distribution of actions is obtained.

To optimize $J(\theta)$, policy gradient algorithms sample trajectories around the current policy π_θ and estimate the parameters θ according to the gradient $\nabla_\theta J(\theta)$ in an ascent manner. This gradient is formulated as:

$$\nabla_\theta J(\theta) = \nabla_\theta \mathbb{E}[\Psi] = \mathbb{E}[\Psi_t \nabla_\theta \log \pi_\theta(a_t|s_t)], \tag{7.1}$$

by using the fact that $\nabla_\theta \log(z) = \frac{1}{z}\nabla_\theta z$. As described by Eq. (7.1), the parameters θ are updated according to the score function Ψ_t, which can take different forms as:

- the state-action value function:

$$Q^{\pi}(s_t, a_t) = \mathbb{E}_{s_{t+1:\infty}, a_{t+1:\infty}} \left[\sum_{l=0}^{\infty} r_{t+l} \right], \tag{7.2}$$

- the advantage function:

$$A^{\pi}(s_t, a_t) = Q^{\pi}(s_t, a_t) - V^{\pi}(s_t), \tag{7.3}$$

  where,

$$V^{\pi}(s_t) = \mathbb{E}_{s_{t+1:\infty}, a_{t:\infty}} \left[\sum_{l=0}^{\infty} r_{t+l} \right], \tag{7.4}$$

- and the Generalized Advantage Estimator (GAE):

$$A^{GAE(\gamma,\lambda)} = \sum_{l=0}^{\infty} (\gamma\lambda)^l \delta_{t+l}^V. \tag{7.5}$$

The state-action value function Q^{π} gives unbiased policy gradient estimates at the cost of high-variance. The advantage function A^{π} uses the state value function V^{π} as a baseline in order to decrease this gradient variance. Similarly, the generalized advantage estimator proposed a combination of weighted TD residuals $\delta_t^V = r_t + \gamma V(s_{t+1}) - V(s_t)$ to establish a balance between bias and variance [Schulman et al., 2016]. This trade off is characterized by a parameter $\lambda \in [0, 1]$, higher values of λ deliver low bias but high variance estimates, and lower values of λ vice versa.

Alternatively, the policy gradient can be estimated by automatic differentiation of an objective function whose gradient is the same as the one given by the policy estimator. One of such objective functions can take the form:

$$L(\theta) = \mathbb{E}_t[A_t \log \pi_\theta(a_t|s_t)]. \tag{7.6}$$

Different surrogate objective function have been demonstrated to work. The Proximal Policy Optimization (PPO) [Schulman et al., 2017] is one of the most widely used RL algorithms, whose surrogate is expressed as:

$$L^{PPO}(\theta) = \mathbb{E}_t[\min(r_t(\theta)A_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)A_t)], \tag{7.7}$$

where, $r_t = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}$ represents the probability ratio, and ϵ is a hyperparameter that defines a bound for the new policy to differ from the previous one. In other words, ϵ serves as a regularizer on the policy by discouraging large policy updates outside the region

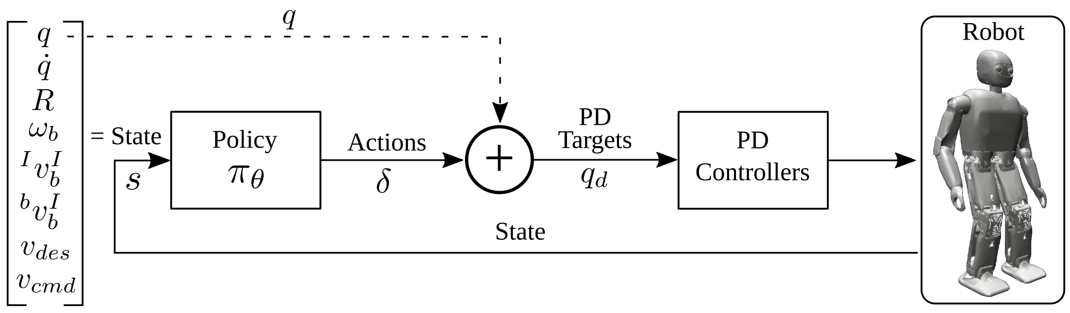

Figure 7.2: Learned walking controller system overview. The control policy π_θ delivers increments δ of the current joint state positions q that ultimately set targets q_d for PD controllers of the actuated joints.

established by ϵ. Both the advantage estimator A_t, and the generalized advantage estimator A^{GAE} can be employed within the PPO algorithm.

The parameterized policy π_θ can be represented by multiple function approximation methods, e.g., by polynomial or by radial basis functions. In practice, however, π_θ is normally parameterized by a neural network. The state value function $V_\phi(s_t)$, needed for the advantage function A^{GAE} computation is also represented by a neural network with parameters ϕ. PPO constitutes an actor-critic method, in which the *critic* is the state value function V_ϕ, and the *actor* is the policy π_θ.

7.3 DRL Problem Definition

The walking controller presented in this chapter provides omnidirectional capabilities to humanoid robots. The complete control system is shown in Fig. 7.2. The controller is fully defined by a control policy π_θ modeled by an actor network (Section 7.3.4). As a conventional learning problem, the approach proposed here incorporates a training and a testing phase. The parameters of the actor network are updated during training in a iterative fashion. First, experiences are collected in a buffer by interacting with the environment, and then, the parameters of the network are updated based on the collected data. After updating the policy network, the buffer is cleared out such that new experiences generated by the updated parameters can be collected. When the training phase finishes, the network parameters are kept fixed and the robot follows the actions dictated by the policy. The formulation of the state space is described in Section 7.3.1. Note that the network does not output directly torque commands but increments δ of PD targets q_d for each of the actuated joints.

7.3.1 State Space

In order to guarantee that the proposed approach can be transferred to real robotic platforms, all elements of the state s need to be observable. This means that ground truth information from the simulator can not be provided to the agent, in contrast with approaches that are employed in the character animation. In other words, the state can only contain data that the robot can infer from sensor measurements.

The state s is composed of: joint positions q; joint velocities $\dot{q}$; orientation of the base of the robot R; the angular velocity of the base ω_b; the linear velocity of the base w.r.t. a inertial reference frame I expressed in I, and in the base link frame b, i.e., ${}^{I}v_b^I$ [2] and ${}^{b}v_b^I$, respectively; a long-term desired velocity v_{des}; and a short-term commanded velocity v_{cmd}. A deeper motivation of each component is provided below.

The joint positions $q \in \mathbb{R}^{n_q}$ and joint velocities $\dot{q} \in \mathbb{R}^{n_q}$ are read directly from the joint encoders for all n_q commanded joints, $R = [R_\alpha, R_\beta, R_\gamma]$ is composed of the roll, pitch and yaw rotation fused angles[3] [Allgeuer and Behnke, 2018] of the base link of the robot. In case yaw estimation is not available, it suffices to provide a two-dimensional R vector containing the roll and pitch orientation.

The angular velocity $\omega_b = [\omega_x, \omega_y, \omega_z] \in \mathbb{R}^3$ is composed of the gyro measurements. The linear velocity of the base link is given by the robot state estimation which relies on the robot kinematics and base link rotation R. The state estimation assumes a flat ground. The inertial reference frame I is initialized at the base link frame and is updated regularly to the current z-axis aligned base link frame. This update serves to clear the accumulated drift error. The linear velocity is expressed on a global ${}^{I}v_b^I \in \mathbb{R}^3$ and local reference frame ${}^{b}v_b^I \in \mathbb{R}^3$. The incorporation of these two vectors is essential for training. If ${}^{I}v_b^I$ is not included in s, the robot might deviate from the original target velocity direction. Small deviations are accumulated in time, which cause large errors as illustrated in Fig. 7.3 (left). In a similar manner, if ${}^{b}v_b^I$ is not incorporated into s, the robot might develop walking patterns that violate the local target velocity. For instance, the robot might learn to walk as displayed in Fig. 7.3 (right).

Finally, the desired target velocity $v_{des} = [v_x^{des}, v_y^{des}, \omega_z^{des}] \in \mathbb{R}^3$ and an interpolated commanded velocity $v_{cmd} = [v_x^{cmd}, v_y^{cmd}, \omega_z^{cmd}] \in \mathbb{R}^3$ are integrated into the state s. This desired velocity v_{des} is the user input or task's goal. This is typically defined by high level planners in autonomous tasks. The interpolated velocity v_{cmd} contributes to smooth velocity transitions when v_{des} changes. The interpolation is implemented by a fixed velocity change every time step until reaching the desired velocity v_{des}. The maximum velocity change in the interpolation defines the mean walking acceleration of the controller.

[2] Notation: for a vector ${}^{A}v_C^B$, the left superscript denotes that the coordinates of the vector are expressed in frame A, for the position of a point C relative to other point B.

[3] Alternative representation of Euler angles

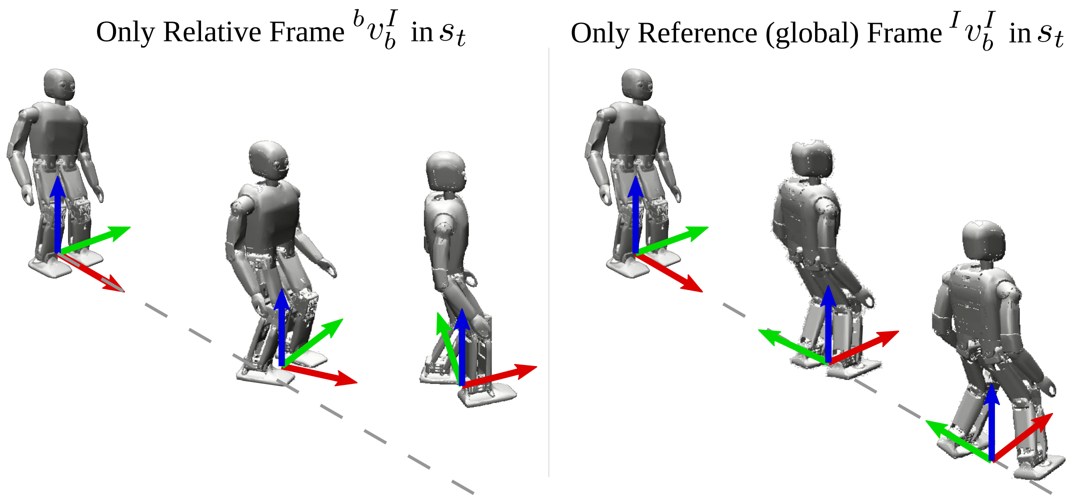
Only Relative Frame ${}^{b}v_{b}^{I}$ in s_t
Only Reference (global) Frame ${}^{I}v_{b}^{I}$ in s_t

Figure 7.3: Commanded velocity representation in state space. Limitations of global ${}^{I}v_{b}^{I}$ and local ${}^{b}v_{b}^{I}$ formulations. Left: a gait learned by a state containing the velocity of the base expressed only on a local frame has drifting problems due to small errors at each foot step. Right: the velocity given only in a local frame might violate the commanded velocity.

7.3.2 Action Space

For learning motor skills for physics-based characters, Peng and Panne [2017] have shown that PD targets results in faster convergence and better performance compared to joint torques or joint velocities. In the approach presented in this chapter, the robot is commanded through PD controllers, but in contrast with [Peng and Panne, 2017], the policy determines PD deltas δ instead of the absolute target values. These delta values together with the current joint position q, define ultimately the target values of the PD controllers $q_d = q + \delta$. The target values are clipped to the joint limits of the robot. Unusual and not anthropomorphic postures such as walking with upright arms or with backwards bent knees are avoided by the joint limits. By learning PD increments, issues arising by abrupt changes in the PD targets are avoided. This is done by setting a maximum bound on the increments δ, which ultimately represent actuator speed limits. These issues might generate, for example, jerky, unnatural motions when the controllers are saturated.

Gaussian policies are the common option in stochastic control. As mentioned in Section 7.2.1, the control policy π_θ determines parameters of probability distributions instead of single deterministic actions. In Gaussian policies, gradients are calculated with respect to the parameters (mean μ and occasionally the variance σ) of a normal distribution which is not bounded. Thus, Gaussian policies are not able to represent the action space as increments, e.g., PD target deltas δ. Therefore, by employing Gaussian policies there is no guarantee to respect the speed actuator limits and to avoid controller saturation. Chou et al. [2017] proposed Beta policies as a manner to bound action spaces. The Beta policies are bounded and have demonstrated faster convergence and higher reward values compared to their Gaussian counterpart. By using Beta policies, the action space is bounded to $\delta \in [\delta_{min}, \delta_{max}]$. The actions are sampled from the beta distribution at 100 Hz. The n_q commanded joints define consequently a n_q-dimensional action space.

7.3.3 Reward Function

The reward function is composed of the following terms: velocity tracking r_{vel}, pose regularization r_{reg}, being alive r_{alive}, and foot clearance r_{foot}. The reward function is formulated as:

$$r = w_v r_{vel} + w_r r_{reg} + w_a r_{alive} + w_f r_{foot}. \tag{7.8}$$

The weights of the reward terms w_* set priorities between the components of the reward function. These priorities are necessary because increasing the reward of one subtask can imply the reduction of another one. For instance, if the robot does not move, it will never fall but it will not track the given target velocity. All reward terms are limited to $r_* \in [0,1] \cdot \Delta t$, where Δt is the policy time step. The limits are implemented by a smooth logistic kernel function $K : \mathbb{R} \to [0,1]$ expressed as $K(x|l) = 2/(e^{lx} + e^{-lx})$. A similar kernel function was proposed in [Hwangbo et al., 2019]. This kernel function is easily parameterized by one single parameter l. High values of l deliver small rewards (more penalization) for a error vector x, while small values of l return higher rewards (less penalization) for the same vector x.

Velocity Tracking

Velocity tracking is the main task of the controller. r_{vel} is formulated as:

$$r_{vel} = C_v \cdot K(e_v)\Delta t,\ e_v = \left\| \begin{bmatrix} v_x^{cmd} - {}^b v_{b,x}^I \\ v_y^{cmd} - {}^b v_{b,y}^I \\ v_x^{cmd} - {}^I v_{b,x}^I \\ v_y^{cmd} - {}^I v_{b,y}^I \\ \omega_z^{cmd} - \omega_z \end{bmatrix} \right\|. \tag{7.9}$$

The velocity error $e_v \in \mathbb{R}^5$ is the difference between the current commanded velocity v^{cmd} and the current 2D-linear and angular velocity of the base of the robot. The linear velocity is expressed in the reference ${}^I v_b^I$ and the local frame ${}^b v_b^I$. The L2-norm contributes to an overall tracking of all velocity components which is not guaranteed when adding up separate kernel functions for each element of e_v. With individual kernel functions, the tracking of some velocity components might affect the tracking of other ones.

In addition to the kernel function applied to the velocity error e_v, this reward term includes a curriculum variable $C \in (0,1]$. This variable changes dynamically the priority of this reward term. At the beginning of training, the priority is low to let the agent learn to stand. The priority is increased rapidly once the robot has a notion of standing. The value of C is defined per epoch: $C_{t+1} = C_t^{k_d}$, where k_d specifies the speed change of C. Without C, the agent would learn a greedy policy in which it tilts to the front, provoking a fall. Once the agent has learned this behavior, it is hard to obtain a more stable one.

Pose Regularization

The policy training process is guided by regularizing a defined robot pose referred to the nominal pose. In this manner, a reference motion is avoided. Without constraining the training to a reference motion simplifies the application of this approach to robots with different kinematics. The dependency on motion capture data is removed with its respective engineered processing. This processing includes synchrony as in [Peng et al., 2018] and kinematic mappings. The nominal pose is defined as a joint configuration (position vector) which can be represented, for instance, as a standing robot pose.

Given the position vector q^r of a set of joints to be regularized, the regularization error $e_{reg} = q^{reg} - q^r$ is formulated as the difference between the nominal pose and q^r. Observe that, if required, some joints are not regularized. The regularization reward term is defined as:

$$r_{reg} = K(\|q^{reg} - q^r\|)\Delta t. \tag{7.10}$$

Alive

The main purpose of this term is to punish when the robot falls, or more precisely, when the robot will inevitably fall. The reward value is determined by the height or z-position, p_z, and the rotation represented by R_α and R_β, of the robot base. A positive reward is given at each time step if these values stay inside certain thresholds: p_z^{min}, R_α^{max} and R_β^{max}, respectively. Beyond these threshold values the robot cannot avoid falling. If the robots falls, the terminal state is reached and the rollout terminates. This term encourages the robot to stay alive for longer time periods, or, in other words, it encourages the robot to maintain balance. Observe that a negative reward given only when the terminal state is reached does not differentiate between shorter and longer walking sequences, rendering the learning problem harder. r_{alive} is defined as:

$$r_{alive} = \begin{cases} \Delta t & \text{if } p_z > p_z^{min}, R_\alpha < R_\alpha^{max}, R_\beta < R_\beta^{max} \\ 0 & \text{else.} \end{cases} \tag{7.11}$$

The alive term plays a key role at the beginning of training when the robot is required to learn to stand and to give the first steps without falling. This is done by encouraging more longer walking sequences. Note that the weight or priority of this term w_{alive} has to be smaller than w_v, in other case, the robot will mostly learn to stand avoiding the risk of falling.

Foot Clearance

The pose regularization reward term does not impose any constraint on the feet clearance with the floor. Policies might be then learned which lift the feet as little as possible. Although such policies show stable walking patterns in simulation, transferred policies to the real robot exhibit, in general, motions with dragging feet which lead to unstable gaits partly caused by model differences and joint backlash. Thus, a feet clearance term is incorporated into the reward function. This term acts on the swing leg only. Apart of considering the clearance ${}^{lf,rf}p_z$, this term includes the orientation ${}^{rf,lf}\phi_x$ of the right (rf) and left (lf) foot to discourage the agent to walk on its lateral feet edges which emerges as an artifact from maximizing the feet clearance ${}^{lf,rf}p_z$. For a right swing leg, for instance, the foot clearance reward term is formulated as:

$$r_{foot} = C_{foot}K(e_{foot})\Delta t, e_{foot} = \left\| \begin{bmatrix} w_\phi {}^{rf}p_z^{des} - {}^{rf}p_z \\ {}^{rf}\phi_x \\ {}^{lf}\phi_x \\ w_\phi {}^{lf}p_z \end{bmatrix} \right\|. \tag{7.12}$$

The first element of e_{foot} encourages the clearance of the swing leg's foot with the floor given a desired clearance value ${}^{rf}p_z^{des}$. The second and third elements (${}^{rf}\phi_x$ and ${}^{lf}\phi_x$) discourage rotations of both feet around the sagittal axis. Finally, the last element aims to keep the contact of the floor with the foot in stance. This element disfavors the development of flight phases, which produce unstable behaviors when the policy is executed on the real robot. The weight w_ϕ is introduced because of the different units of e_{foot}, namely meters and radians. For a left swing leg, the superscripts of Eq. (7.12) are interchanged correspondingly. The swing leg is defined as the leg whose foot has the smallest distance to the trunk link in the z axis. Hysteresis is added when changing the swing leg to discourage flight phases.

7.3.4 Actor and Critic Networks

As explained in Section 7.2.1, the approach presented in this chapter renders an actor-critic method. The actor is the control policy π_θ, while the critic estimates the state value function required to calculate the generalized advantage estimator A^{GAE}. Both, the actor and the critic are modeled by neural networks which need to be trained. In inference, however, only the actor network is employed. The parameters ϕ of the critic network are training by optimizing the following loss function:

$$L^V = (V_\phi(s_t) - \hat{V})^2, \tag{7.13}$$

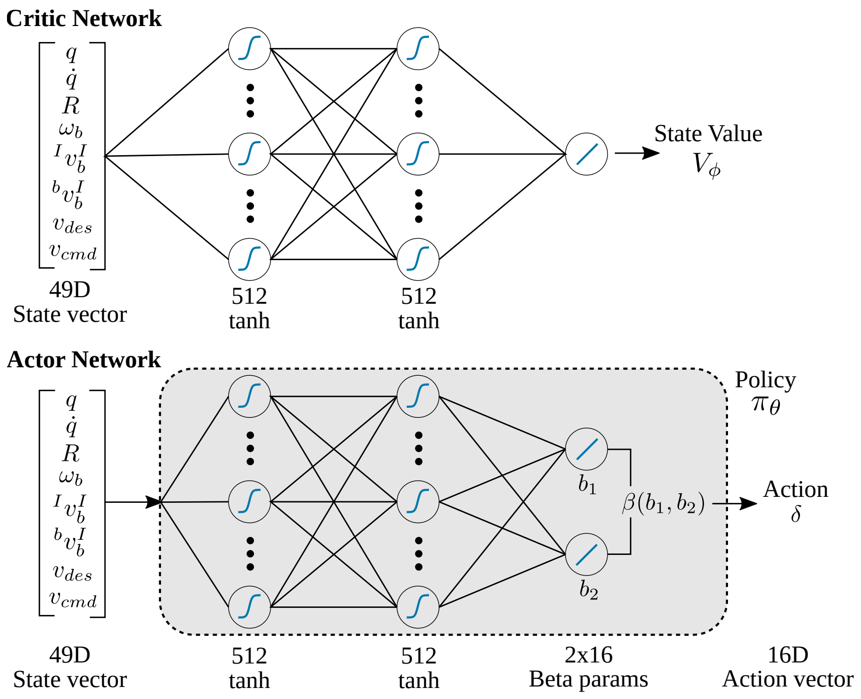

Figure 7.4: Critic and actor network architectures. The input vector is defined by the state s_t. Both networks have two fully-connected layers with 512 units with $\tanh$ activations. The critic network delivers the state value V_ϕ, whereas the actor outputs the action $a_t = \delta$ (PD targets) depending on the inferred b_1 and b_2 parameters of the beta distributions.

where $\hat{V}$ represents the sampled state value calculated from collected experiences (trajectories). The actor network weights are updated by maximizing the PPO loss function (Eq. 7.7).

Figure 7.4 shows the critic and actor networks, whose architectures have several similarities. Each of them has two hidden fully-connected layers of 512 units with $\tanh$ activation functions. The input layer is fully-connected to the state vector s_t while the last layer differs for both. For the critic network, a single unit estimates the state value V_ϕ. The last layer of the actor network, uses two units for each commanded joint. These two units represent the b_1 and b_2 parameters of the Beta distributions. During training, the actions are sampled from the beta distributions while in inference the actions are defined by the distribution mode.

7.3.5 Curriculum Learning for Target Velocities

Inspired by human learning systems, curriculum learning [Bengio et al., 2009] has been widely used in reinforcement learning as a manner to guide the training process by gradually

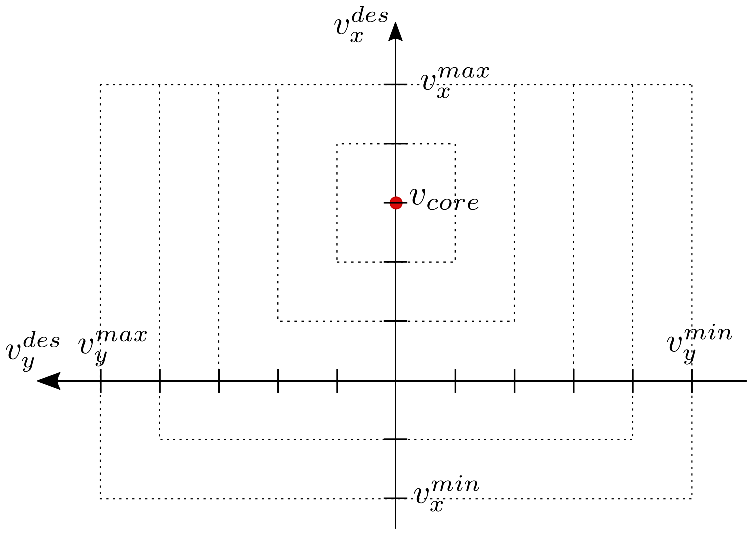

Figure 7.5: Velocity scheduler as curriculum strategy. The region from where target velocities are sampled is enlarged gradually. The bounds of the sampling region are clipped by $v_{x,y}^{max}$ and $v_{x,y}^{max}$. For the sake of clarity, w_z is not shown.

increasing the task difficulty. Thus, complex tasks are learned by fulfilling a series of easier tasks first. The omnidirectional capabilities presented in this chapter are learned by proposing a novel curriculum strategy.

This curriculum strategy is implemented as a velocity scheduler that increases the task difficulty gradually. The velocity scheduler defines the bounds from where a target velocity $v^{des} \in \mathbb{R}^3$ is sampled each episode from a three-dimensional uniform distribution. In the first episode, the robot is commanded to walk only in the sagittal direction at a fixed velocity v_{core}. The bounds of the regions, from where the target velocities are sampled, are gradually increased as training progresses. We define an episode ζ, in which the target velocities stop increasing, The bound values for each episode are linearly interpolated according to the episode ζ and the maximum target velocities. The training process continues until the maximum number of episodes is reached. In this manner, the agent has more time to be asked to walk at maximum velocities, and at the same time, the agent can refine the learned policy.

The velocity scheduler increases the task difficulty smoothly, in contrast with large fixed velocity increments at a periodic rate. This smooth transition is helpful in avoiding local minima. Large increments of the target velocity bounds might slow down the overall training process, because the agent is reluctant to change the policy that has been delivering high velocity tracking rewards.

Figure 7.5 illustrates the curriculum strategy, i.e., the regions from which target velocities for v_x and v_y are sampled. For clarity in the Figure, angular velocities are not shown.

The introduction of the core velocity $v_{core} \gg [0, 0, 0]$ is motivated by several reasons. First, the robot does not need to learn to walk forwards and backwards from scratch simultaneously which is in general a more complex task. Second, it encourages more walking behaviors than standing ones And, finally, it discourages the robot to learn to slide instead of walking.

7.3.6 Training

The control policy is trained using PPO and GAE. Experiences are collected in parallel in multiple environments to speed up the training time. Each episode starts with the robot in the standing position. The actions are sampled from the beta distributions at each time step. The target velocity v_{des} is given by the velocity scheduler presented in Section 7.3.5. An episode ends when a fixed number of time steps are reached.

7.4 Real world transfer

In the last years, remarkable improvements to physics-based simulation has paved the path for utilizing robot learning approaches. The most widely used simulators include: MuJoCo [4], PyBullet [5], and PhysX [6]. Although the fidelity of the simulation in general has been improved, there are still several differences between the virtual and the real world, which constitute the *reality gap*.

Transferring walking controllers from simulation to the real robot is a challenging task for several reasons. The high complexity of the real world is difficult to model: real objects are not perfect; object properties vary according to ambient conditions; and the modeling of all elements of an entire robotic systems (including cables and screws) is unpractical from a computationally point of view. In the same manner, to reduce the reality gap by considering material properties and their interactions in a microscopic level is computationally very costly, and thus, this microscopic modeling is not suitable for robot learning approaches, as the one presented here. Other factors that affect the sim-to-real transfer include: noisy sensor input data and the complexity to model the actuation. Gehring et al. [2016], for example, proposed a model for a real SAE actuator which contains nearly 100 parameters. A few of those parameters can be obtained from datasheets but the rest need to be defined by experiments. This process can take several weeks even with similar actuation models available in the literature.

In this section, we introduce several strategies that facilitate the sim-to-real transfer by increasing the robustness of the policy to dissimilarities between the virtual and the real world. These strategies include: system identification, noise injection, dynamics

[4] http://mujoco.org/

[5] https://pybullet.org

[6] https://developer.nvidia.com/physx-sdk

randomization, the modeling of actuation latency, and network output filtering. Below, these strategies are explained in detail:

- **System identification**. The identification mainly contributes to find a good initial set of simulation parameters. These parameters affect naturally the learning process and therefore the obtained behaviors. For instance, low friction coefficients of the floor would resemble slippery surfaces such as ice, which requires different motions compared to a rigid floor.

  We found that the rotor, also known as, reflected inertia is a decisive parameter for learning a stable gait, which corresponds with the observations also made by Xie et al. [2019]. Specifically, we notice that low values of this parameter lead to jerky motions that are very challenging to control for the agent.

  Furthermore, tuning the PD controllers of the real robot and the simulation to get similar responses in both environments is critical. Because the responses of the PD controllers influence the joint state configuration directly, and therefore the network input, a single untuned PD controller suffices to produce a previously non-observed input that might lead to continually increasing instabilities. For instance, if the response of a certain joint controller is faster in the real world, that joint will probably overshoot the expected position which the network will compensate in the next cycle and might result in a second overshoot (of the compensation). As result, a jerky motion will emerge. This problem is amplified with the number of controllers whose response is different than the one used for training.

- **Noise injection**. Additional noise is purposely injected to the output network in order to address the issues described above, namely the problems associated with dissimilar responses of the PD controllers between simulation and real robot. For each joint, the introduced noise is sampled from a uniform distribution $\eta_{pd} \sim \mathcal{U}(1 - \epsilon_{pd}, 1 + \epsilon_{pd})$. η_{pd} acts as a scale factor of the inferred actions, i.e., of the inferred target deltas δ. The noised PD targets are then defined as:

$$q_d = q + \eta_{pd} \cdot \delta. \tag{7.14}$$

  At the beginning of each episode, noise is independently sampled for each joint. The noise values are not changed during the episode. For values $\eta_{pd} > 1$, a faster response of PD controllers is modeled, while values $\eta_{pd} < 1$ represent a slower response. Because the noise is randomly sampled at each episode, the agent is exposed to different PD responses every episode, encouraging the agent to be robust against different controller responses.

  To model real sensory data, noise is also added to the sensors in simulation, namely to the gyroscope (η_g), the accelerometer (η_a), the joint position encoders (η_q) and joint velocity sensors (η_v). In this manner, all the elements of the state s, i.e., the network input, that depend on sensory input data contain simulated noise. The added sensory noises are independently randomly sampled from zero-mean Gaussian distributions $\mathcal{N}(0, \eta_{g,a,q,v})$.

- **Dynamics randomization**. Parameters such as frictions, masses and inertia matrices have a big impact on the dynamics of the simulation. Parameters related to the robot model (masses, inertias and joint positions) are obtained directly from CAD. We have observed that the friction model plays a key role on the sim-to-real transfer. The tangential (lateral) friction is relevant for avoiding slippage mostly when the robot is commanded to move in the sagittal and lateral directions. The torsional friction (opposition to rotation around the contact normal) regulates turning motions. Finally, we found that the rolling friction (opposition to rotating around the contact tangent) needs to be incorporated into the friction model. Without modeling the rolling friction, some stable gaits in simulation could not be transferred to the real platform because initial contacts of the heels with the floor resulted in unstable gait patterns, in which the robot tried to walk on its heels only. By modeling rolling friction, such walking patterns are discouraged during training. The friction between two surfaces is modeled by elliptic cones. All friction coefficients (tangential μ_t, torsional μ_z and rolling μ_r) are randomly sampled from uniform distributions $\mu_{t,z,r} \sim \mathcal{U}(\mu_{t,z,r}^{min}, \mu_{t,z,r}^{max})$ at the beginning of each episode. In this manner, each walking sequence defines different friction parameters.

- **Modeling of actuation latency**. We define latency as the time an actuator takes to read its actual position after a commanded target has been sent. In simulation, this happens immediately. However, with the real robot, this delay implies that the learned policy is taking actions on states that do not fully represent the actual state of the robot. In this manner, the robot constantly overshoots the desired joint configuration learned in simulation, which leads to unstable gaits and jerky motions, which results in falling.

  Following the approach proposed by Tan et al. [2018], we model actuation latency by keeping a history of observations and by feeding previous observations to the network instead of the current ones according to a delay time t_{lat}. Observations are recorded at the same frequency of the control policy. The network input is then defined as a linear interpolation between adjacent observations according to the latency $t_i \leq t_{lat} \leq t_{i+1}$. To make the policy robust against this latency value, a latency value t_{lat} is uniformly sampled $t_{lat} \sim \mathcal{U}(t_{lat}^{min}, t_{lat}^{max})$ at the beginning of each episode. The incorporation of the latency model was one of the most important factors in successfully transferring the gait to the real platform.

- **Network output filtering**. The locomotion control dictated by a learned policy is in general prone to instabilities coming from different sources including noisy sensory input data, non-perfect tuning of PD controllers, actuation latencies, and simulation parameter mismatches. These instabilities translate to a noisy network output. We filter the action values inferred by the policy before sending the corresponding commands to the actuators using a butterworth low-pass filter (cutoff frequency of 10 Hz in our experiments).

- **Observations and commanded targets bounds** The real-world observations are bounded to the values used in training, before passing them to the network as input. In addition, the commanded joint targets are clipped by the joint limits.

7.5 Evaluation

The approach presented here is evaluated on the NimbRo-OP2X humanoid robot [Ficht et al., 2018]. The platform presents a parallel kinematics structure in each leg and its parts are fully 3D printed, including the transmission gears. In total, the robot has 18 DOFs, five in each leg, three per arm, and two (pan and tilt) for controlling the head. The latter two joints, however, are not included in the state and the action spaces, i.e., they are not controlled by the learned walking controller, because their contribution to walking is considered negligible and these joints are usually controlled by visual perception modules. The action space is then 16-dimensional. Due to the parallel kinematics, the legs do not posses an ankle pitch joint. The relative position between the shanks and feet are determined by the hip and knee joint positions. The leg kinematics is composed by: hip roll, hip pitch, hip yaw, knee pitch and ankle roll joints. The state estimation of the robot only delivers a 2D orientation (roll and pitch angles) of the base link. The state space is then composed of 49 dimensions.

The walking sequences are performed in the MuJoCo physics-based simulator. The PD controller and the simulation run at the same frequency, i.e., 1 kHz. The control policy, on the other hand, makes an inference each 25 ms, setting a control frequency of 40 Hz. In total, 12 environments run in parallel on an Intel i9-9900K CPU. The task is implemented as an Open Gym environment[Brockman et al., 2016]. The networks are written in PyTorch and the PPO algorithm was based on an open source implementation[Kostrikov, 2018]. The gym environment (coded in Python) is connected to the simulation (written in C++) using the ROS middleware.

For each network update, each environment collects experiences in 800 time steps, thus, with 12 environments, the networks are trained with 9600 time steps per epoch. The batch size of both networks is 480 with a learning rate of 1×10^{-4}. The Adam optimizer is employed with 10 updates per epoch. The learning decay factor is set to $\gamma = 0.99$ and the bias-variance parameter of the GAE is set to $\tau = 0.97$. The networks are trained for 7,400 episodes which corresponds to a total of 7.1×10^7 time steps. The total training time equals 32.5 hours which corresponds to 20.5 days of simulated (virtual) time.

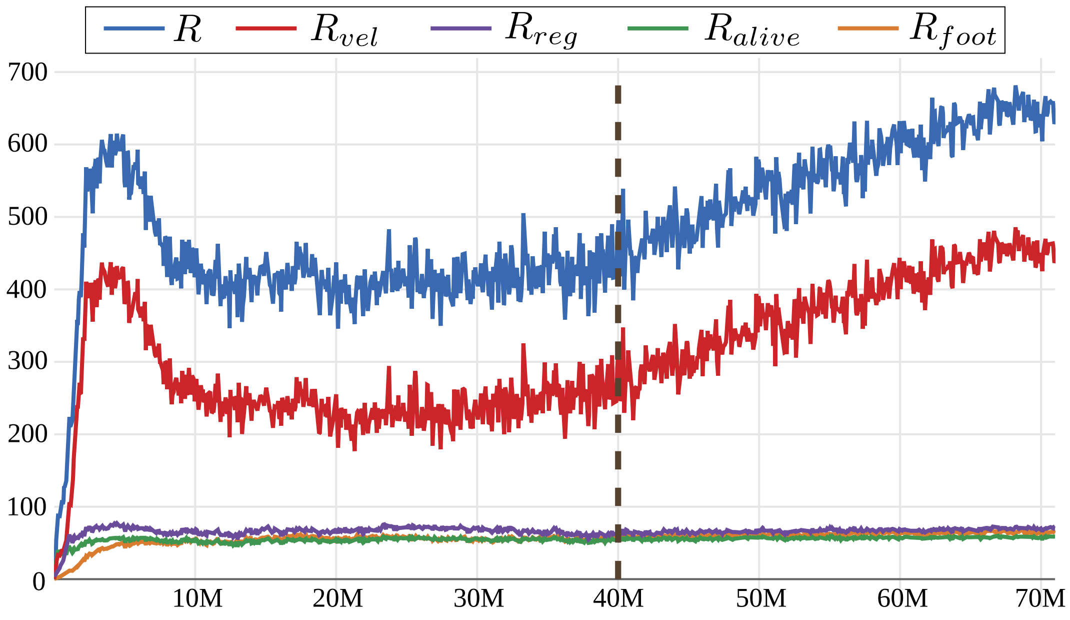

Figure 7.6: Training return curves. The vertical dashed line represents the point where the limits of the velocity scheduler have been reached. Note how the agent continues learning and refining the policy after the limits of the velocity scheduler have been reached.

The weights of the reward terms are set manually from experience: $w_v = 42$, $w_r = 4$, $w_a = 4$ and $w_{foot} = 18$. The velocity tracking and pose regularization kernels are parameterized with $l_v = 9$, $l_r = 3$, and $l_{foot} = 10$ respectively. The harder kernel function on the velocity tracking means that the tracking error needs to be smaller than the pose regularization error to yield a similar reward. The softer constraint on the pose discourages rigid poses.

The velocity scheduler (curriculum strategy) is set as follows. The robot is asked to learn to walk at $v_{core} = [0.4\,\mathrm{m/s}, 0.0\,\mathrm{m/s}, 0.0\,\mathrm{rad/s}]$. The minimum and maximum velocities in each direction are bounded to $v_x \in [-0.6, 0.6]$, $v_y \in [-0.6, 0.6]$, and $\omega_z \in [-0.6, 0.6]$. Finally, the curriculum variables of the reward term are initialized at $C_v = 0.01$ with $k_d = 0.95$, and $C_{foot} = 0.05$ with $k_d = 0.995$.

Figure 7.6 shows the reward training curves. The individual contribution of all reward terms are also displayed. The vertical dash line indicates when the maximum target velocities are reached. The robot learns to avoid falling after 2.5×10^5 time steps. This is evidenced by a alive reward higher than 60, which results in a non-falling rate higher 75% (alive maximum value is 80). At the end of training, the robot falls rarely having an alive reward value oscillating around 80. Falls happen mostly when robot is commanded to the velocity limits.

Qualitative results are presented in Figure 7.7. The omnidirectional capabilities are demonstrated by showing the robot performing multiple walking maneuvers. At the top

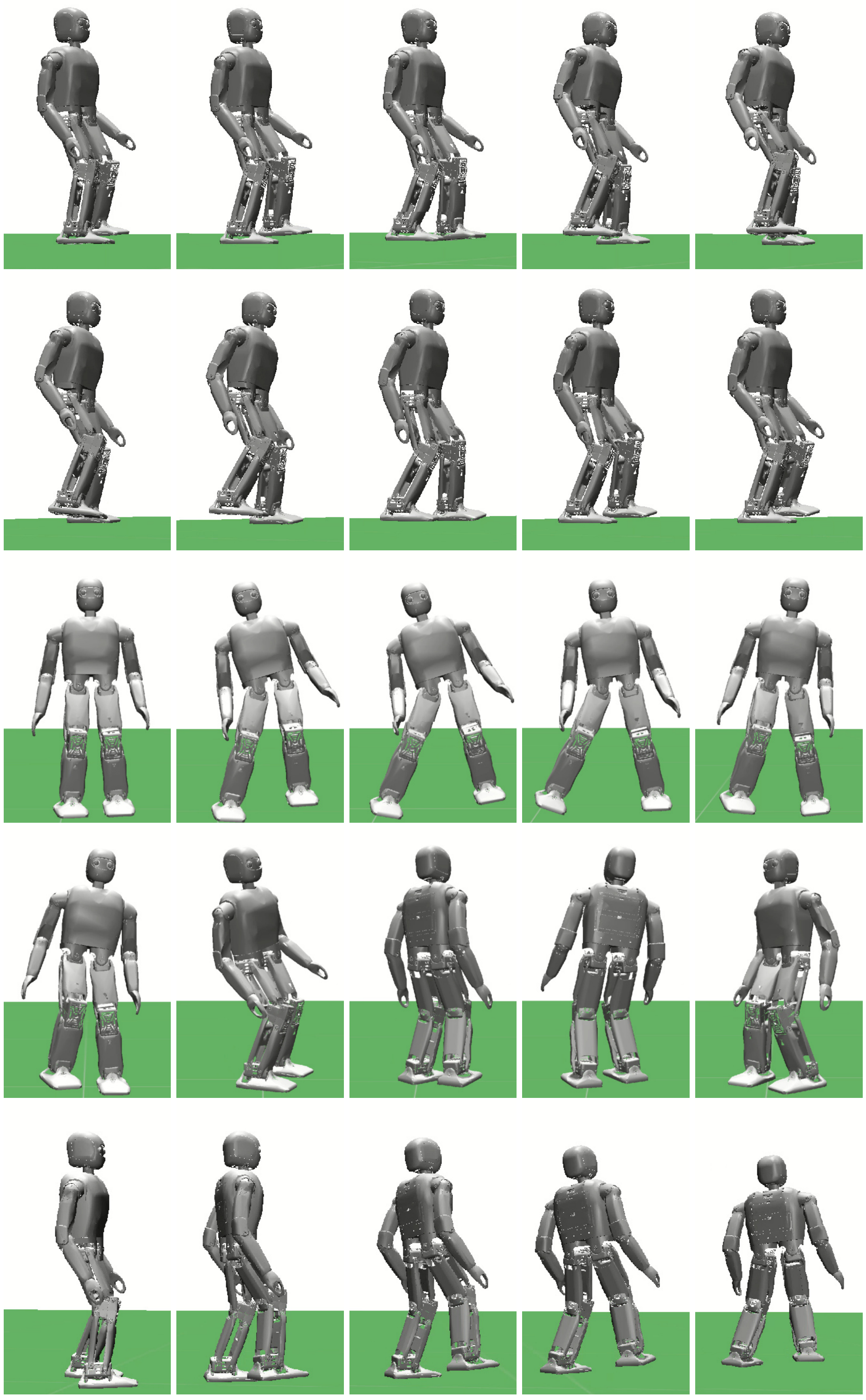

Figure 7.7: Snapshots of omnidirectional motions executed by the learned walking controller. From top to bottom: forward at 0.6 m/s, backward at 0.5 m/s, lateral at 0.6 m/s, turning at 0.5 rad/s and forward with turning at (0.6 m/s, 0.2 rad/s).

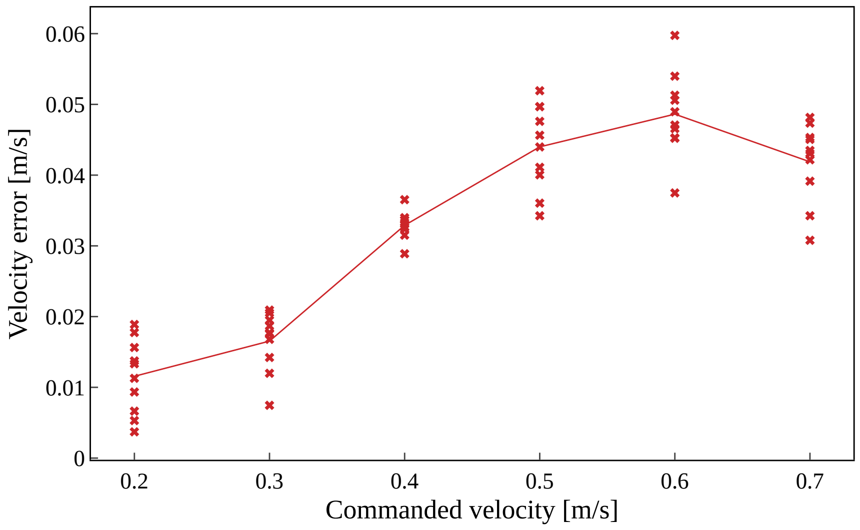

Figure 7.8: Velocity tracking error with respect to commanded velocities.

and second rows, the NimbRo-OP2X walks forwards and backwards, respectively. In the third row the robot walks in the lateral plane, while in the fourth row the humanoid is shown tuning on the spot. Finally, in the last row, the robot turns left during forward walking. Another maneuvers not displayed include: diagonal walking, turning during backward walking, and turning during lateral walking. The most complex maneuver achieved by our learned walking controller is to turn during diagonal walking. In summary, the learned controller is able to produce fully omnidirectional patterns by generating walking motions in each individual and combined directions: sagittal, lateral and turning. Interestingly, the robot learns to walk in place, i.e., to lift the feet rhythmically without moving in any direction.

The velocity tracking capabilities of the walking controller are exhibited in Figure 7.9. Solid lines represent the given desired velocities while dashed lines show the measured velocities. The robot is asked to walk in place and after to start walking forward at 0.3 m/s. Then, the velocity is increased to 0.6 m/s. Later, backward and sideways walking is requested at low (0.15 m/s) and high (0.5 m/s) speeds. This is followed by turning on the spot left and right at 0.5 rad/s. Next, the humanoid is commanded to walk in combined directions: diagonal in all four possibilities and turning during forward and lateral walking.

The average tracking error of walking sequences dictated by forward walking commands at different speeds is presented in Figure 7.8. For each commanded velocity, ten walking sequences are performed with a duration of 10 seconds each.

The robustness of the learned controller is also evaluated. Walking sequences of 60 seconds are carried out for multiple target velocities. For each commanded velocity, ten sequences are performed. The number of falls once the sequence is terminated is counted and presented in Table 7.1.

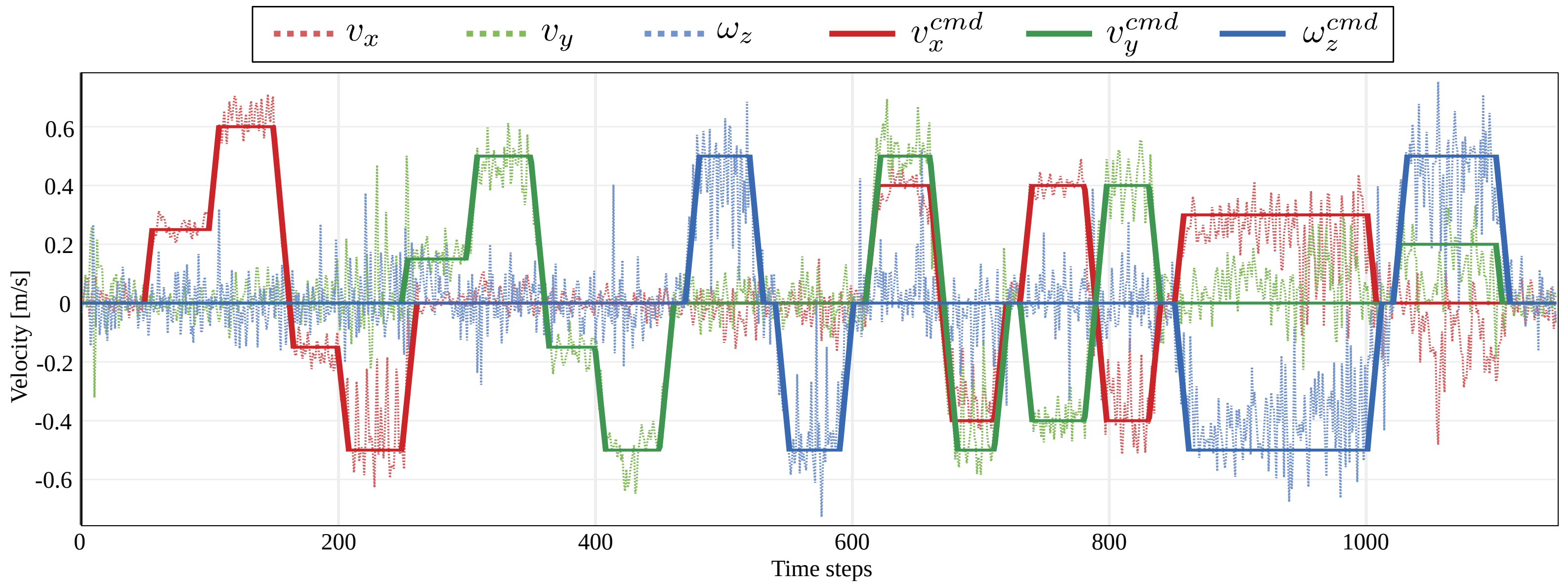

Figure 7.9: Velocity tracking given desired target velocities. Multiple consecutive target velocities (solid wider lines) in different directions with varying velocities are tracked by the learned controller. The robot is initially requested to walk in place and then in individual directions (forward, backward, left, right and turning). Then, motions with combined directions (sagittal with lateral, sagittal with turning and lateral with turning) are evaluated.

Table 7.1: Results of falling test

Commanded Velocity	# Falls	Commanded Velocity	# Falls
$v = [0.0, 0.0, 0.0]$	0/10	$v = [0.0, 0.0, 0.6]$	1/10
$v = [0.6, 0.0, 0.0]$	0/10	$v = [0.0, 0.0, -0.6]$	3/10
$v = [-0, 6, 0.0, 0.0]$	2/10	$v = [0.4, 0.4, 0.0]$	0/10
$v = [0.0, 0.6, 0.0]$	0/10	$v = [0.4, 0.3, 0.0]$	0/10
$v = [0.0, -0.6, 0.0]$	0/10	$v = [0.0, 0.4, 0.3]$	0/10

During the test presented in Table 7.1, the robot has only fallen with high speeds in the backward directions and turning. This can be explained by the fact, that the number of experiences the robot had walking backwards is less than their forward counterpart. With respect to turning, the robot is evaluated at its limits. An additional test is performed to evaluate the capability of the controller to manage velocity changes. For this, target velocities are sampled inside the velocity bounds and 10 seconds walking sequences are carried out. The sequences are performed consecutively. From 160 velocity changes, the controller managed successfully 150, resulting in a 93.75% success rate.

Push-recovery experiments are also carried out with the learned controller. Forces from the front, the back and the side are exerted individually at the base link frame for 0.2 s when the humanoid is commanded to different target velocities. The initial perturbation equals 10 N. After 10 pushes, the exerted force is incremented by 10 N until the robot can not recover at least once out of the 10 pushes. Table 7.2 presents the maximum push the robot is able to recover from each of the directed perturbations. Furthermore, the maximum force that the robot is able to recover successfully for all 10 pushes is also given in Table 7.2. This latter test is called 100% Success.

The contribution of the velocity scheduler is evaluated by training a walking controller without the scheduler. After the same number of epochs and same parameters, the robot was not able walk in any direction, it stands without moving. This highlights the complexity of the task and the impossibility to learn it without a curriculum as guidance in the current control regime. Moreover, in an additional experiment, the Beta policy is replaced by a Gaussian one. This altered controller is not able to make more than 4 consecutive steps without falling due to the iterative saturation of the PD controllers. This last experiments underlines the importance of Beta policies on the use of energy and on the avoidance of the incorporation of torque terms in the reward function, its corresponding measurement or estimation and weighting in the reward function.

Transfer to real robot

Finally, the learned controller is transferred to the real robot. As described in Section 7.4, a proper tuning of the PD controllers is performed before starting the training in simulation.

Table 7.2: Results gait perturbation test. Values expressed in [N].

Commanded velocity	Test	Front push	Back push	Lateral p. right	Lateral p. left
In-place	100% Success	40	20	50	
	Max. push	60	30	90	
$v = [0.3, 0, 0]$	100% Success	30	20	50	
	Max. push	50	30	80	
$v = [0.6, 0, 0]$	100% Success	30	10	40	
	Max. push	50	20	70	
$v = [0, 0.25, 0]$	100% Success	40	20	40	40
	Max. push	60	30	60	100
$v = [0, 0.5, 0]$	100% Success	40	20	20	50
	Max. push	60	30	60	100

Small adjustment of the PD values of the real controllers are also made to match the response of the learned gait. The injected noise to the response of PD controllers η_{pd} is set to 0.1. The standard deviations of the zero-mean Gaussian noise applied to the sensory data take the following values: $\eta_g = 1 \times 10^{-4}$, $\eta_a = 1 \times 10^{-4}$, $\eta_q = 1 \times 10^{-3}$, and $\eta_v = 1 \times 10^{-3}$. Furthermore, the friction values are sampled from $\mu_t \sim \mathcal{U}(0.4, 0.8)$, $\mu_z \sim \mathcal{U}(0.1, 0.3)$, and $\mu_r \sim \mathcal{U}(0.0, 0.2)$, for the tangential, torsional, and rolling frictions, respectively. Naturally, the injected noise and sampled friction coefficients are considered only in simulation. The latency t_{lat} is sampled uniformly from the range $[0, 50]$ms in simulation, and it is set to a measured value of $t_{lat} = 8$ms for the real robot. Finally, the cutoff frequency of the low-pass filter is set to 10 Hz.

The learned control policy is executed successfully on the real robot. Snapshots of the robot walking forward, backward, laterally and turning on spot are presented from Fig. 7.10 to Fig. 7.13. Interestingly, the robot is able to walk omnidirectionally, although the motions executed with the real robot do not match perfectly with the motions performed in simulation, This suggests a further improvement on the sim-to-real transfer approach presented in this chapter.

7.6 Discussion

In this chapter, a walking controller is learned from experiences gathered in simulation. The approach does not have any notion or previous knowledge about gait generation, i.e., no central pattern generators or body dynamics are employed. The motions are completely

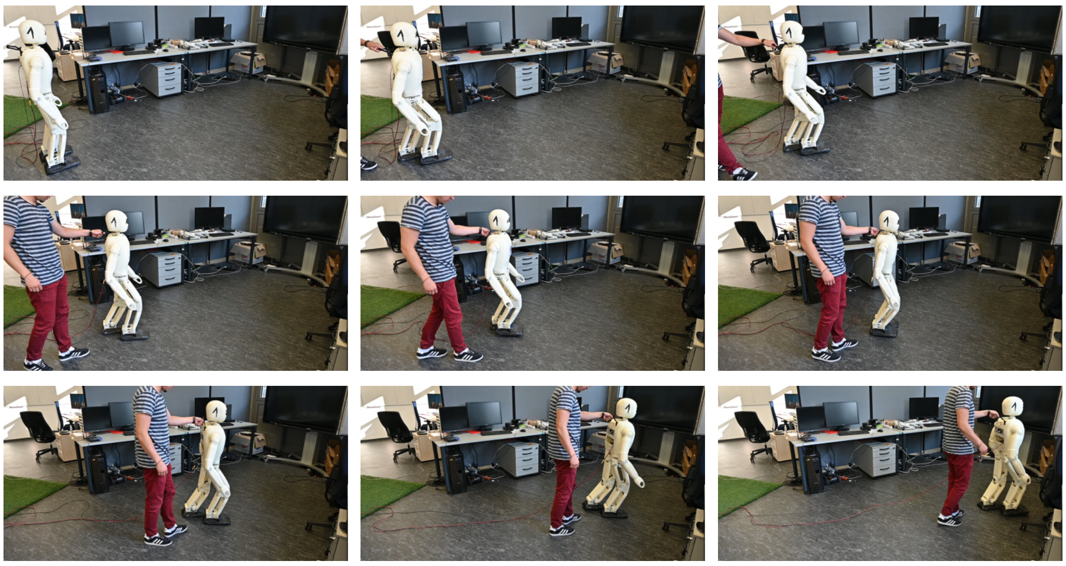

Figure 7.10: Snapshots of forward walk performed by the real robot commanded by the learned locomotion controller.

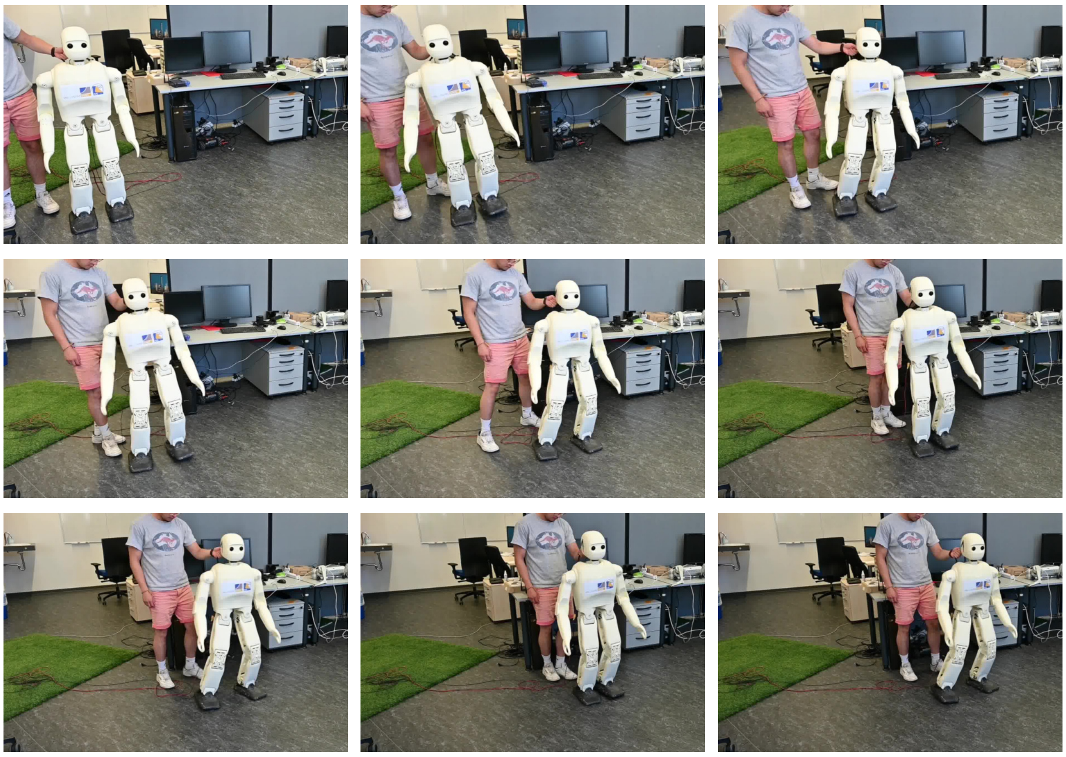

Figure 7.11: Snapshots of lateral (left) walk performed by the real robot commanded by the learned locomotion controller.

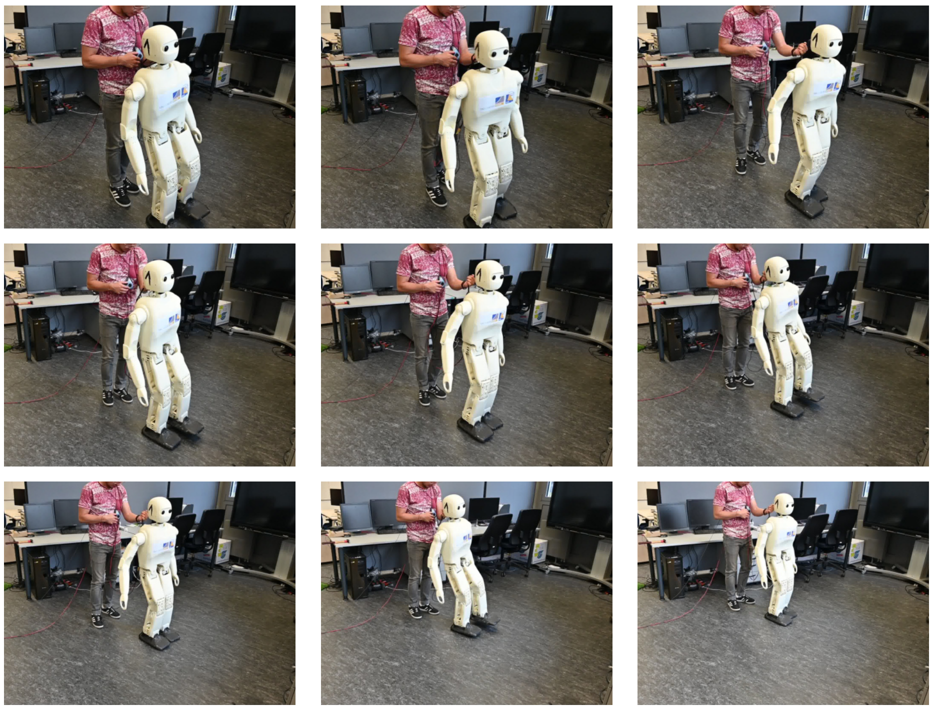

Figure 7.12: Snapshots of backward walk performed by the real robot commanded by the learned locomotion controller.

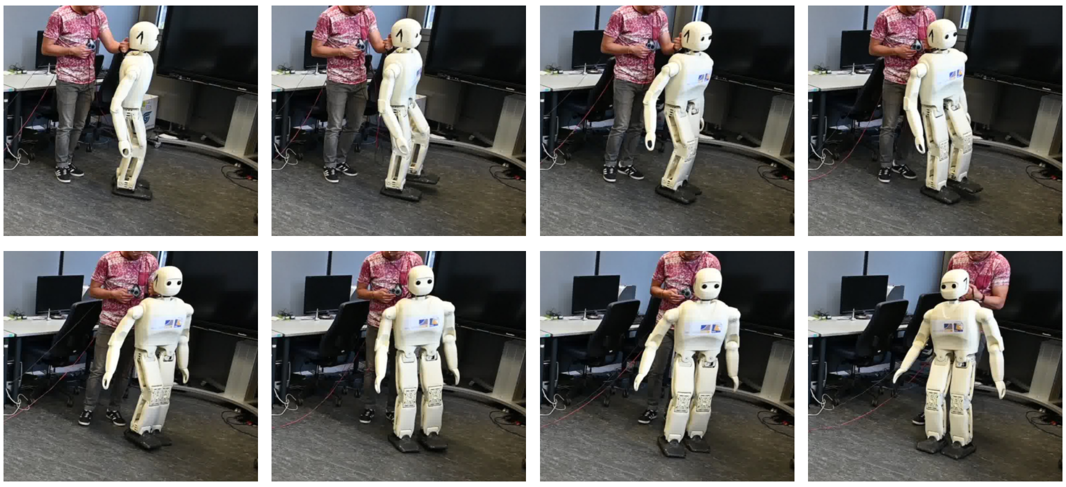

Figure 7.13: Snapshots of backward walk performed by the real robot commanded by the learned locomotion controller.

learned. The developed walking controller exhibits omnidirectional capabilities allowing a humanoid robot to move forward, backwards, laterally, diagonal, to turn or to turn during sagittal or lateral walking. This is possible thanks to the introduction of a velocity scheduler as a curriculum strategy. In this manner, the robot is not directly faced with a complex task but it reaches its goal by gradually increasing the difficulty of simpler tasks. Furthermore, the developed learned controller does not require reference motions, which increases its applicability to different robot kinematics and removes the dependency on the availability of motion capture data or reference motion acquisition in general. Our approach leverages on a single nominal pose instead.

The wide repertoire of learned locomotion behaviors is achieved by a single policy, i.e., a single neural network. This facilitates velocity changes because of shared parameters. In addition, individual training and further integration of separate policies is avoided. Moreover, it was demonstrated how the use of Beta policies can effectively be implemented to bound the action space and correspondingly to define elegantly actuator limits.

As future research directions, learning a robust gait able to handle different kind of terrains, such as those with rough or slippery surfaces, will increase the applicability of such approaches into real-world applications. Moreover, learning 3D walking controllers is an exciting but complex task that poses challenges such as 3D balance and state estimation. This would allow the robot to climb stairs and go through slope terrains, facilitating the deployment of bipedal robots in human scenarios.

8 Conclusion

"There is no real ending. It's just the place where you stop the story."

— Frank Herbert

In this thesis, we presented different novel learning approaches that enable humanoid robots to perform grasping and walking motions.

Initially, an approach was proposed that registers novel instances of an object category in a non-rigid manner. The registration was formulated as a learning approach that learns typical geometrical variations. The acquired knowledge enabled us to reconstruct the observed objects recovering unseen parts that are relevant for robotics applications such as grasp planning. We have evaluated our method on different object categories with partially observed data. We showed that the developed shape registration method is robust against noise coming from real sensory data and pose misalignments. Interestingly, our method is able to register objects online based on single-view observations, and thus, it can be implemented in several real-world tasks.

For applications where the use of 3D depth sensors is not feasible or not desired by design, we have developed a novel approach for category-level non-rigid registration based on single-view RGB images. We established that a Convolutional Neural Network (CNN) can infer deformations on the visible parts of the observed object. This validates our hypothesis that 3D deformations can be inferred employing similar network architectures typically used for optical flow. Moreover, we demonstrated how objects can be reconstructed by incorporating a learned shape space of the object category. Our approach was evaluated on synthetic and real RGB images outperforming the Coherent Point Drift (CPD) method even without depth information and with misalignments of the object pose. Our learned models were trained based on rendered images only, but they can infer successfully 3D deformations of real objects. This corroborates the application of rendering approaches for real-world applications but it also underlines the importance of realistic textured models. As result of the non-rigid registration on RGB images, objects which are hard to perceive by depth sensors (e.g., those with transparent surfaces) were successfully registered.

Then, an approach was presented that infers grasping motions by transferring associated knowledge from a canonical instance to the observed objects based on the shape space registration developed previously. In other words, a canonical grasping motion is transformed according to the geometry of the observed instance. This transfer allowed us to generate grasping motions for novel instances online thanks to the knowledge encoded in shape spaces of object categories. Interestingly, the grasp motion inference does not require specialized computational devices, and thus, the inference can be performed onboard, which enables the approach to be deployed in several robotic platforms. The grasping transfer was integrated into complete pipelines for autonomous functional grasping of previously unseen objects of known categories. The evaluation of the approach was carried out with synthetic and real sensory data in single and dual-arm experiments. We showed that our approach was able to efficiently generate functional grasping motions enabling a potential object usage. Moreover, we demonstrated that the grasping transfer is agnostic to the robot kinematics, and thus, it was implemented in different robotic platforms with little effort.

Later, the grasping transfer was enriched by considering multiple successful grasping experiences. Consequently, a transfer model was trained associating shape descriptors with grasping motions. Our method was able to transfer grasping skills with a real robotic platform from experiences collected only in simulation. All the computations were performed online and run onboard, which demonstrates the feasibility and practicability of our approach to be deployed on real robotic systems.

Regarding multi-fingered robot hands, we have developed a novel approach for generating grasp hand configurations based on the object shape. The geometry is represented as a shape descriptor computed as result of a shape space registration, while the hand configurations are described by postural synergies. We have shown that object shapes can be associated with grasp configurations in a supervised manner using Gaussian Processes. We demonstrated that Gaussian distributions can be employed to represent the multiple ways a particular object can be grasped functionally. The effectiveness of this method was evaluated both in simulation and in the real robot controlling nine actuated degrees of freedom. Interestingly, large variations in the object geometries could be successfully handled for simple geometries, which paves the path for implementing this method with more complex object shapes.

In the second part of this thesis, we first presented an approach to trade off simulations and real-robot experiments for optimizing gait parameters. The experiments were combined in a Bayesian optimization framework, that selects the most informative points to be evaluated based on the relative entropy. We showed how the gait stability of a real humanoid robot was improved with the parameters found by our approach. Thanks to the incorporation of simulation experiments in the framework, the system wear off of the robot is decreased due to the reduced number of physical real experiments. For the performed optimization, the real robot did not fall a single time, which demonstrates the generalization capabilities of our method from simulation to the real robot.

Finally, we have developed a novel approach to learn a single control policy capable of omnidirectional walking using a realistic humanoid robot model. We have shown the capacity of the learned policy to walk in the sagittal and lateral directions and to turn around the vertical axis at different speeds. Without altering the policy, and in an online manner, our approach produces motions in combined directions, i.e., the agent is able to walk diagonal and to turn during walking. The achievement of these locomotion patterns was possible mainly due to: a velocity scheduler, that dynamically changes the limits where target velocities are sampled; the introduction of a core velocity to start the training; the use of beta policies to bound the action space and to incorporate actuation limits in the learning problem; and the introduction of a nominal pose to regularize and to guide the training of the agent. Interestingly, the proposed learning method does not require reference motions and achieves anthropomorphic locomotion. We followed a minimalist approach to design the reward function by defining only four reward terms.

The learned omnidirectional locomotion controller has been transferred to the real hardware. The sim-to-real transfer was possible without real robot experiments for training. System identification, noise injection, dynamics randomization, and actuation latency modeling are the key factors that significantly contributed to achieve the transfer.

Outlook and Future Work

The methods presented in this thesis open several directions for future research. The shape space registration can be extended to handle more complex objects that are composed by several parts. Similar to the global registration presented in this thesis, a part-wise registration can be incorporated for each individual object part. In this manner, a holistic registration can infer the global object shape and the finer details are captured by the individual registration of each part. In addition, the backbone method of the shape registration, i.e., the Coherent Point Drift, could be replaced by more robust approaches such as [Ma et al., 2016].

An interesting open problem in the shape registration is considering color surfaces or textures. This could contribute to the automatic generation of new textured instances. The texture registration is also relevant for teleoperated applications especially for the visualization of reconstructed objects.

In real world applications, the category-level registration from RGB images needs to be integrated with other perception modules to infer the category class and the object pose. Thus, the presented CNN can be extended to infer additionally, pixel-wise object categories to better match the masks used for training. Moreover, the incorporation of an object pose refinement module on the network and the shape space could increase robustness against misalignments.

An alternative approach to infer 3D deformations of objects observed in RGB images is differentiable rendering. The incorporation of shape spaces could infer deformations of the non-observable parts of the objects, similar as the approach presented in this thesis. A comparison of our shape registration from RGB images with an approach based on differentiable rendering could provide more insight into the problem.

Regarding the final grasp performed by a multi-fingered hand, additional sensory modalities (e.g., joint currents and force-torque sensors) can be exploited and incorporated into the grasping transfer model. Thus, a grasp inference could be described by a postural configuration and a compliance value. The introduction of the compliance could play a key role for grasping delicate objects such as vegetables, eggs, etc. Additionally, non-linear methods for dimensionality reduction such as GP-LVM, have shown lower reconstructions errors compared to its linear counterpart [Romero et al., 2013]. The construction of synergy spaces using these kind of methods could yield better results and consequently a better grasp pose transfer.

We have observed limitations on the approach that combines simulation and real robot experiments for gait parameter optimization with higher dimensions. A promising alternative to address these limitations is the use of dimensionality reduction methods and to perform the optimization in a lower-dimensional space exploiting the relations of some joints during walking, e.g., the right arm moves synchronously with the left leg.

Learned walking controllers for humanoid robots still have to overcome several challenges to be implemented in daily life scenarios. Even though several common components of analytic gaits such as central pattern generators, gait planners, dynamic models, and trajectory planners can be efficiently learned, a considerable tuning effort has to be made especially for the reward functions. An automatic and dynamic self weight (importance) balancing is an exciting research direction. Efficient representations of gait styles (or motion regularizers) could also enrich the presented approach. Learning to walk in 3D is still an open interesting problem to be tackled, allowing humanoid robots to walk on sloped terrains and to climb stairs. An additional consideration of multiple terrains such as grass or those with slippery surfaces will also contribute to the state of the art in learning of locomotion controllers.

Finally, the integration of manipulation and walking capabilities into a single approach, that allows to generate whole body motions to interact with objects in the environment, is an interesting research direction based on the work presented on this thesis. Tasks such as lifting objects from the floor, opening doors with whole body motions, kicking a ball during walking, among others, could contribute to the generation of more anthropomorphic motions and to the overall goal of deploying humanoid robots into quotidian scenarios.

List of Figures

List of Tables

List of Videos

Acronyms

CNN	Convolutional Neural Network
COM	Center Of Mass
CPD	Coherent Point Drift
CPG	Central Pattern Generator
CPU	Central Processing Unit
CSR	Category-Level Shape Registration
DOF	Degrees Of Freedom
DRL	Deep Reinforcement Learning
EM	Expectation Maximization
ES	Entropy Search
GAE	Generalized Advantage Estimator
GMM	Gaussian Mixture Model
GP	Gaussian Process
GPU	Graphics Processing Unit
ICP	Iterative Closest Point
IMU	Inertial Measurement Unit
MF-ES	Multi-Fidelity Entropy Search
MLP	Multi-Layer Perceptron
PCA	Principal Component Analysis
PPO	Proximal Policy Optimization
RBF	Radial Basis Function
RGB	Red, Green and Blue (image)
RGB-D	Red, Green, Blue and Depth (image)
RL	Reinforcement Learning
RNN	Recurrent Neural Network
ROS	Robot Operating System

Bibliography

Agarwal, S., K. Mierle, and Others (2012). *Ceres Solver*. `http://ceres-solver.org`.

Akrour, R., D. Sorokin, J. Peters, and G. Neumann (2017). "Local Bayesian optimization of motor skills". In: *Proceedings of the 34th International Conference on Machine Learning (PMLR)*.

Allen, B., B. Curless, and Z. Popović (2003). "The space of human body shapes: reconstruction and parameterization from range scans". In: *ACM Transactions on Graphics (TOG)*, pp. 587–594.

Allgeuer, P. and S. Behnke (2016). "Omnidirectional bipedal walking with direct fused angle deedback mechanisms". In: *International Conference on Humanoid Robots (Humanoids)*.

Allgeuer, P. and S. Behnke (2018). "Fused angles and the deficiencies of Euler angles". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*.

Allgeuer, P., H. Farazi, G. Ficht, M. Schreiber, and S. Behnke (2016). "The igus Humanoid Open Platform: A child-sized 3D printed open-source robot for research". In: *Künstliche Intelligenz*.

Amor, H. B., O. Kroemer, U. Hillenbrand, G. Neumann, and J. Peters (2012). "Generalization of human grasping for multi-fingered robot hands". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 2043–2050.

Balasubramanian, R., L. Xu, P. D. Brook, J. R. Smith, and Y. Matsuoka (2012). "Physical human interactive guidance: Identifying grasping principles from human-planned grasps". In: *IEEE Transactions on Robotics (T-RO)*, pp. 899–910.

Behnke, S. (2006). "Online trajectory generation for omnidirectional biped walking". In: *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*.

Bengio, Y., J. Louradour, R. Collobert, and J. Weston (2009). "Curriculum learning". In: *Proceedings of the 26th International Conference on Machine Learning (ICML)*. ACM, pp. 41–48.

Berkenkamp, F., A. P. Schoellig, and A. Krause (2016). "Safe controller optimization for quadrotors with Gaussian processes". In: *IEEE International Conference on Robotics and Automation (ICRA)*.

Bernardino, A., M. Henriques, N. Hendrich, and J. Zhang (2013). "Precision grasp synergies for dexterous robotic hands". In: *IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pp. 62–67.

Bicchi, A. and V. Kumar (2000). "Robotic grasping and contact: A review". In: *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 348–353.

Blanz, V. and T. Vetter (1999). "A morphable model for the synthesis of 3D faces". In: *Proceedings of the 26th Conference on Computer Graphics and Interactive Techniques (SIGGRAPH)*. ACM Press/Addison-Wesley Publishing Co., pp. 187–194.

Bohg, J., A. Morales, T. Asfour, and D. Kragic (2013). "Data-driven grasp synthesis—a survey". In: *IEEE Transactions on Robotics*.

Brockman, G., V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba (2016). *OpenAI Gym*. eprint: `arXiv:1606.01540`.

Burghard, O., A. Berner, M. Wand, N. Mitra, H.-P. Seidel, and R. Klein (2013). "Compact part-based shape spaces for dense correspondences". In: *CoRR* abs/1311.7535.

Calandra, R., N. Gopalan, A. Seyfarth, J. Peters, and M. Deisenroth (2014). "Bayesian gait optimization for bipedal locomotion". In: *Proceedings of the 8th International Conference on Learning and Intelligent Optimization (LION)*.

Carr, J. C., R. K. Beatson, J. B. Cherrie, T. J. Mitchell, W. R. Fright, B. C. McCallum, and T. R. Evans (2001). "Reconstruction and representation of 3D objects with radial basis functions". In: *Proceedings of the 28th Conference on Computer Graphics and Interactive Techniques (SIGGRAPH)*. ACM, pp. 67–76.

Chang, A. X., T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, J. Xiao, L. Yi, and F. Yu (2015). *ShapeNet: An information-rich 3D model repository*. Tech. rep. Stanford University — Princeton University — Toyota Technological Institute at Chicago.

Chou, P.-W., D. Maturana, and S. Scherer (2017). "Improving stochastic policy gradients in continuous control with deep reinforcement learning using the beta distribution". In: *Proceedings of the 34th International Conference on Machine Learning (ICML)*, pp. 834–843.

Choy, C. B., D. Xu, J. Gwak, K. Chen, and S. Savarese (2016). “3D-R2N2: A unified approach for single and multi-view 3D object reconstruction”. In: *European Conference on Computer Vision (ECCV)*. Springer, pp. 628–644.

Chui, H. and A. Rangarajan (2003). “A new point matching algorithm for non-rigid registration”. In: *Computer Vision and Image Understanding*, pp. 114–141.

Ciocarlie, M. T. and P. K. Allen (2009). “Hand posture subspaces for dexterous robotic grasping”. In: *The International Journal of Robotics Research*, pp. 851–867.

Dai, A., C. Ruizhongtai Qi, and M. Nießner (2017). “Shape completion using 3D-encoder-predictor CNNs and shape synthesis”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5868–5877.

Dame, A., V. A. Prisacariu, C. Y. Ren, and I. Reid (2013). “Dense reconstruction using 3D object shape priors”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1288–1295.

Deisenroth, M. P., D. Fox, and C. E. Rasmussen (2015). “Gaussian processes for data-efficient learning in robotics and control”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Dempster, A. P., N. M. Laird, and D. B. Rubin (1977). “Maximum likelihood from incomplete data via the EM algorithm”. In: *Journal of the Royal Statistical Society: Series B (Methodological)*, pp. 1–22.

Detry, R., C. H. Ek, M. Madry, J. Piater, and D. Kragic (2012). “Generalizing grasps across partly similar objects”. In: *2012 IEEE International Conference on Robotics and Automation*. IEEE, pp. 3791–3797.

Diankov, R. (2010). “Automated construction of robotic manipulation programs”. In: *Ph.D. Thesis*.

Ekvall, S. and D. Kragic (2007). “Learning and evaluation of the approach vector for automatic grasp generation and planning”. In: *Proceedings IEEE International Conference on Robotics and Automation (ICRA)*.

Engelmann, F., J. Stückler, and B. Leibe (2016). “Joint object pose estimation and shape reconstruction in urban street scenes using 3D shape priors”. In: *German Conference on Pattern Recognition (GCPR)*, pp. 219–230.

Fankhauser, P., M. Bloesch, D. Rodriguez, R. Kaestner, M. Hutter, and R. Siegwart (2015). “Kinect v2 for mobile robot navigation: Evaluation and modeling”. In: *International Conference on Advanced Robotics (ICAR)*, pp. 388–394.

Farazi, H., F. Ahmadinejad, F. Maleki, and M. Shiri (2013). "Evolutionary approach for developing fast and stable offline humanoid walk". In: *3rd Joint Conference of AI and Robotics and 5th RoboCup Iran Open International Symposium*.

Farchy, A., S. Barrett, P. MacAlpine, and P. Stone (2013). "Humanoid robots learning to walk faster: from the real world to simulation and back". In: *Proceedings of the International Conference on Autonomous Agents and Multi-agent Systems (AAMAS)*.

Faria, D. R., P. Trindade, J. Lobo, and J. Dias (2014). "Knowledge-based reasoning from human grasp demonstrations for robot grasp synthesis". In: *Robotics and Autonomous Systems*.

Feix, T., J. Romero, H. B. Schmiedmayer, A. M. Dollar, and D. Kragic (2016). "The GRASP taxonomy of human grasp types". In: *IEEE Transactions on Human-Machine Systems*, pp. 66–77.

Ficht, G., H. Farazi, A. Brandenburger, D. Rodriguez, D. Pavlichenko, P. Allgeuer, M. Hosseini, and S. Behnke (2018). "NimbRo-OP2X: Adult-sized open-source 3D printed humanoid robot". In: *IEEE-RAS 18th International Conference on Humanoid Robots (Humanoids)*.

Ficht, G., H. Farazi, D. Rodriguez, D. Pavlichenko, P. Allgeuer, A. Brandenburger, and S. Behnke (2020). "NimbRo-OP2X: Affordable adult-sized 3D-printed open-source humanoid robot for research". In: *International Journal of Humanoid Robotics (IJHR)*.

Ficuciello, F., D. Zaccara, and B. Siciliano (2016a). "Synergy-based policy improvement with path integrals for anthropomorphic hands". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1940–1945.

Ficuciello, F., G. Palli, C. Melchiorri, and B. Siciliano (2014). "Postural synergies of the UB Hand IV for human-like grasping". In: *Robotics and Autonomous Systems*.

Ficuciello, F., D. Zaccara, and B. Siciliano (2016b). "Learning grasps in a synergy-based framework". In: *International Symposium on Experimental Robotics*, pp. 125–135.

Gabardi, M., M. Solazzi, D. Leonardis, and A. Frisoli (2018). "Design and evaluation of a novel 5 DoF underactuated thumb-exoskeleton". In: *IEEE Robotics and Automation Letters (RA-L)*, pp. 2322–2329.

Gehring, C., S. Coros, M. Hutter, C. D. Bellicoso, H. Heijnen, R. Diethelm, M. Bloesch, P. Fankhauser, J. Hwangbo, M. Hoepflinger, et al. (2016). "Practice makes perfect: An optimization-based approach to controlling agile motions for a quadruped robot". In: *IEEE Robotics & Automation Magazine*.

Hahnel, D., S. Thrun, and W. Burgard (2003). "An extension of the ICP algorithm for modeling nonrigid objects with mobile robots". In: *18th International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 915–920.

Hanna, J. P. and P. Stone (2017). "Grounded action transformation for robot learning in simulation". In: *Association for the Advancement of Artificial Intelligence Conference (AAAI)*.

Hasler, N., C. Stoll, M. Sunkel, B. Rosenhahn, and H.-P. Seidel (2009). "A statistical model of human pose and body shape". In: *Computer Graphics Forum*. Wiley Online Library, pp. 337–346.

Heess, N., S. Sriram, J. Lemmon, J. Merel, G. Wayne, Y. Tassa, T. Erez, Z. Wang, S. Eslami, M. Riedmiller, et al. (2017). "Emergence of locomotion behaviours in rich environments". In: *CoRR abs/1707.02286*.

Heijmink, E., A. Radulescu, B. Ponton, V. Barasuol, D. G. Caldwell, and C. Semini (2017). "Learning optimal gait parameters and impedance profiles for legged locomotion". In: *IEEE-RAS 17th International Conference on Humanoid Robots (Humanoids)*.

Hengst, B., M. Lange, and B. White (2011). "Learning ankle-tilt and foot-placement control for flat-footed bipedal balancing and walking". In: *11th IEEE-RAS International Conference on Humanoid Robots (Humanoids)*.

Hennig, P. and C. Schuler (2012). "Entropy search for information-efficient global optimization". In: *Journal of Machine Learning Research*.

Horaud, R., F. Forbes, M. Yguel, G. Dewaele, and J. Zhang (2010). "Rigid and articulated point registration with expectation conditional maximization". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 587–602.

Huang, Q.-X., G.-X. Zhang, L. Gao, S.-M. Hu, A. Butscher, and L. Guibas (2012). "An optimization approach for extracting and encoding consistent maps in a shape collection". In: *ACM Transactions on Graphics (TOG)*, p. 167.

Huang, S. H., J. Pan, G. Mulcaire, and P. Abbeel (2015). "Leveraging appearance priors in non-rigid registration, with application to manipulation of deformable objects". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 878–885.

Huebner, K., K. Welke, M. Przybylski, N. Vahrenkamp, T. Asfour, D. Kragic, and R. Dillmann (2009). "Grasping known objects with humanoid robots: A box-based approach". In: *International Conference on Advanced Robotics (ICAR)*.

Hwangbo, J., J. Lee, A. Dosovitskiy, D. Bellicoso, V. Tsounis, V. Koltun, and M. Hutter (2019). "Learning agile and dynamic motor skills for legged robots". In: *Science Robotics*.

Ilg, E., N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy, and T. Brox (2017). "Flownet 2.0: Evolution of optical flow estimation with deep networks". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2462–2470.

Kazhdan, M. and H. Hoppe (2013). "Screened poisson surface reconstruction". In: *ACM Transactions on Graphics (ToG)*, p. 29.

Khoury, A. E., F. Lamiraux, and M. Taix (2013). "Optimal motion planning for humanoid robots". In: *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3136–3141.

Kim, V. G., Y. Lipman, and T. Funkhouser (2011). "Blended intrinsic maps". In: *ACM Transactions on Graphics (TOG)*, p. 79.

Klamt, T., D. Rodriguez, L. Baccelliere, X. Chen, D. Chiaradia, T. Cichon, M. Gabardi, P. Guria, K. Holmquist, M. Kamedula, H. Karaoguz, N. Kashiri, A. Laurenzi, C. Lenz, D. Leonardis, E. Mingo, L. Muratore, D. Pavlichenko, F. Porcini, Z. Ren, F. Schilling, M. Schwarz, M. Solazzi, F. Michael, A. Frisoli, M. Gustmann, P. Jensfelt, K. Nordberg, J. Rossmann, U. Suess, N. Tsagarakis, and S. Behnke (2019a). "Flexible disaster response of tomorrow - Final presentation and evaluation of the CENTAURO system". In: *IEEE Robotics and Automation Magazine (RAM), Special Issue on Humanoid Robot Applications in Real World Scenarios*.

Klamt, T., D. Rodriguez, M. Schwarz, C. Lenz, D. Pavlichenko, D. Droeschel, and S. Behnke (2018). "Supervised autonomous locomotion and manipulation for disaster response with a centaur-like robot". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*.

Klamt, T., M. Schwarz, C. Lenz, L. Baccelliere, D. Buongiorno, T. Cichon, A. DiGuardo, D. Droeschel, M. Gabardi, M. Kamedula, N. Kashiri, A. Laurenzi, D. Leonardis, L. Muratore, D. Pavlichenko, A. Periyasamy, D. Rodriguez, M. Solazzi, A. Frisoli, M. Gustmann, J. Rossmann, U. Suess, N. Tsagarakis, and S. Behnke (2019b). "Remote mobile manipulation with the centauro robot: Full-body telepresence and autonomous operator assistance". In: *Journal of Field Robotics (JFR)*.

Kostrikov, I. (2018). *PyTorch Implementations of Reinforcement Learning Algorithms*. `https://github.com/ikostrikov/pytorch-a2c-ppo-acktr-gail`.

Krebs, J., T. Mansi, H. Delingette, L. Zhang, F. C. Ghesu, S. Miao, A. K. Maier, N. Ayache, R. Liao, and A. Kamen (2017). "Robust non-rigid registration through agent-based action learning". In: *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer, pp. 344–352.

Kroemer, O. B., R. Detry, J. Piater, and J. Peters (2010). "Combining active learning and reactive control for robot grasping". In: *Robotics and Autonomous systems*, pp. 1105–1116.

Li, H. and Y. Fan (2017). "Non-rigid image registration using fully convolutional networks with deep self-supervision". In: *arXiv:1709.00799*.

Li, Y., G. Wang, X. Ji, Y. Xiang, and D. Fox (2018). "DeepIm: Deep iterative matching for 6D pose estimation". In: *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 683–698.

Ma, J., J. Zhao, and A. L. Yuille (2016). "Non-rigid point set registration by preserving global and structures". In: *IEEE Transactions on Image Processing*, pp. 53–64.

Marco, A., F. Berkenkamp, P. Hennig, A. P. Schoellig, A. Krause, S. Schaal, and S. Trimpe (2017). "Virtual vs. real: Trading off simulations and physical experiments in reinforcement learning with Bayesian optimization". In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*.

Marco, A., P. Hennig, J. Bohg, S. Schaal, and S. Trimpe (2016). "Automatic LQR tuning based on Gaussian process global optimization". In: *IEEE International Conference on Robotics and Automation (ICRA)*.

Marsland, S., C. J. Twining, and C. J. Taylor (2003). "Groupwise non-rigid registration using polyharmonic clamped-plate splines". In: *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer, pp. 771–779.

Missura, M. and S. Behnke (2013). "Self-stable omnidirectional walking with compliant joints". In: *8th Workshop on Humanoid Soccer Robots*. International Conference on Humanoid Robots (Humanoids).

Morales, A., T. Asfour, P. Azad, S. Knoop, and R. Dillmann (2006). "Integrated grasp planning and visual object localization for a humanoid robot with five-fingered hands". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 5663–5668.

Myronenko, A. and X. Song (2010). "Point set registration: Coherent point drift". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, pp. 2262–2275.

Nealen, A., T. Igarashi, O. Sorkine, and M. Alexa (2006). "Laplacian mesh optimization". In: *Proceedings of the 4th International Conference on Computer graphics and interactive techniques in Australasia and Southeast Asia*, pp. 381–389.

Nguyen, A., M. Ben-Chen, K. Welnicka, Y. Ye, and L. Guibas (2011). "An optimization approach to improving collections of shape maps". In: *Computer Graphics Forum*. Wiley Online Library, pp. 1481–1491.

Ovsjanikov, M., Q. Mérigot, F. Mémoli, and L. Guibas (2010). "One point isometric matching with the heat kernel". In: *Computer Graphics Forum*. Wiley Online Library, pp. 1555–1564.

Pavlichenko, D., D. Rodriguez, C. Lenz, M. Schwarz, and S. Behnke (2019). "Autonomous bimanual functional regrasping of novel object class instances". In: *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*.

Pavlichenko, D., D. Rodriguez, M. Schwarz, C. Lenz, A. S. Periyasamy, and S. Behnke (2018). "Autonomous dual-arm manipulation of familiar objects". In: *IEEE-RAS 18th International Conference on Humanoid Robots (Humanoids)*.

Peng, X. B., P. Abbeel, S. Levine, and M. van de Panne (2018). "Deepmimic: Example-guided deep reinforcement learning of physics-based character skills". In: *ACM Transactions on Graphics (TOG)*, p. 143.

Peng, X. B., G. Berseth, K. Yin, and M. Van De Panne (2017). "Deeploco: Dynamic locomotion skills using hierarchical deep reinforcement learning". In: *ACM Transactions on Graphics (TOG)*, p. 41.

Peng, X. B. and M. van de Panne (2017). "Learning locomotion skills using DeepRL: Does the choice of action space matter?" In: *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. ACM, p. 12.

Periyasamy, A. S., M. Schwarz, and S. Behnke (2019). "Refining 6D object pose predictions using abstract Render-and-Compare". In: *IEEE-RAS 19th International Conference on Humanoid Robots (Humanoids)*.

Rai, A., R. Antonova, S. Song, W. C. Martin, H. Geyer, and C. G. Atkeson (2018). "Bayesian optimization using domain knowledge on the ATRIAS biped". In: *IEEE International Conference on Robotics and Automation (ICRA)*.

Ren, Z., C. Zhou, S. Xin, and N. Tsagarakis (2017). "HERI hand: A quasi dexterous and powerful hand with asymmetrical finger dimensions and under actuation". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 322–328.

Rock, J., T. Gupta, J. Thorsen, J. Gwak, D. Shin, and D. Hoiem (2015). "Completing 3D object shape from one depth image". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2484–2493.

Rodriguez, D. and S. Behnke (2018). "Transferring category-based functional grasping skills by latent space non-rigid registration". In: *IEEE Robotics and Automation Letters (RA-L)*.

Rodriguez, D., C. Cogswell, S. Koo, and S. Behnke (2018a). "Transferring grasping skills to novel instances by latent space non-rigid registration". In: *IEEE International Conference on Robotics and Automation (ICRA)*.

Rodriguez, D. and S. Behnke (2021). "DeepWalk: Omnidirectional bipedal gait by deep reinforcement learning". In: *IEEE International Conference on Robotics and Automation (ICRA), to appear*.

Rodriguez, D., A. Brandenburger, and S. Behnke (2018b). "Combining simulations and real-robot experiments for Bayesian optimization of bipedal gait stabilization". In: *Proceedings of 22nd RoboCup International Symposium, Montreal, Canada*.

Rodriguez, D., A. Di Guardo, A. Frisoli, and S. Behnke (2018c). "Learning postural synergies for categorical grasping through shape space registration". In: *IEEE-RAS 18th International Conference on Humanoid Robots (Humanoids)*.

Rodriguez, D., H. Farazi, P. Allgeuer, G. Ficht, D. Pavlichenko, A. Brandenburger, J. Kürsch, and S. Behnke (2018d). "Advanced soccer skills and team play of RoboCup 2017 TeenSize winner NimbRo". In: *RoboCup 2017: Robot World Cup XXI*.

Rodriguez, D., F. Huber, and S. Behnke (2020). "Category-level 3D non-rigid registration from single-view RGB images". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*.

Röfer, T. (2005). "Evolutionary gait-optimization using a fitness function based on proprioception". In: *RoboCup 2004: Robot Soccer World Cup VIII*.

Romero, J., T. Feix, C. H. Ek, H. Kjellström, and D. Kragic (2013). "Extracting postural synergies for robotic grasping". In: *IEEE Transactions on Robotics*, pp. 1342–1352.

Ruehl, S. W., C. Parlitz, G. Heppner, A. Hermann, A. Roennau, and R. Dillmann (2014). "Experimental evaluation of the Schunk 5-Finger gripping hand for grasping tasks". In: *IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pp. 2465–2470.

Santello, M., M. Flanders, and J. F. Soechting (1998). "Postural hand synergies for tool use". In: *Journal of Neuroscience*.

Sarac, M., M. Solazzi, E. Sotgiu, M. Bergamasco, and A. Frisoli (2017). “Design and kinematic optimization of a novel underactuated robotic hand exoskeleton”. In: *Meccanica*, pp. 749–761.

Schulman, J., P. Moritz, S. Levine, M. Jordan, and P. Abbeel (2016). “High-dimensional continuous control using generalized advantage estimation”. In: *Proceedings of the International Conference on Learning Representations (ICLR)*.

Schulman, J., F. Wolski, P. Dhariwal, A. Radford, and O. Klimov (2017). “Proximal policy optimization algorithms”. In: *CoRR abs/1707.06347*.

Schwarz, M. and S. Behnke (2020). “Stillleben: realistic scene synthesis for deep learning in robotics”. In: *IEEE International Conference on Robotics and Automation (ICRA)*.

Schwarz, M., C. Lenz, G. M. García, S. Koo, A. S. Periyasamy, M. Schreiber, and S. Behnke (2018). “Fast object learning and dual-arm coordination for cluttered stowing, picking, and packing”. In: *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3347–3354.

Shoemake, K. (1992). “Uniform random rotations”. In: *Graphics Gems*. Morgan Kaufmann, pp. 124–132.

Stouraitis, T., U. Hillenbrand, and M. A. Roa (2015). “Functional power grasps transferred through warping and replanning”. In: *IEEE International Conference on Robotics and Automation (ICRA)*.

Stueckler, J., R. Steffens, D. Holz, and S. Behnke (2011). “Real-time 3D perception and efficient grasp planning for everyday manipulation tasks”. In: *European Conference on Mobile Robots (ECMR)*, pp. 177–182.

Stulp, F., E. Theodorou, M. Kalakrishnan, P. Pastor, L. Righetti, and S. Schaal (2011). “Learning motion primitive goals for robust manipulation”. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*.

Tan, J., T. Zhang, E. Coumans, A. Iscen, Y. Bai, D. Hafner, S. Bohez, and V. Vanhoucke (2018). “Sim-to-real: Learning agile locomotion for quadruped robots”. In: *Robotics: Science and Systems (RSS)*.

Tevs, A., M. Bokeloh, M. Wand, A. Schilling, and H.-P. Seidel (2009). “Isometric registration of ambiguous and partial data”. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1185–1192.

Tipping, M. E. and C. M. Bishop (1999). “Probabilistic principal component analysis”. In: *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, pp. 611–622.

Tobin, J., R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel (2017). "Domain randomization for transferring deep neural networks from simulation to the real world". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*.

Tremblay, J., T. To, and S. Birchfield (2018a). "Falling things: A synthetic dataset for 3D object detection and pose estimation". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition workshops (ICVPR)*, pp. 2038–2041.

Tremblay, J., T. To, B. Sundaralingam, Y. Xiang, D. Fox, and S. Birchfield (2018b). "Deep object pose estimation for semantic robotic grasping of household objects". In: *Conference on Robot Learning (CoRL)*.

Tsounis, V., M. Alge, J. Lee, F. Farshidian, and M. Hutter (2019). "DeepGait: Planning and control of quadrupedal gaits using deep reinforcement learning". In: *CoRR abs/1909.08399*.

Vahrenkamp, N., L. Westkamp, N. Yamanobe, E. E. Aksoy, and T. Asfour (2016). "Part-based grasp planning for familiar objects". In: *16th IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, pp. 919–925.

Wang, H., S. Sridhar, J. Huang, J. Valentin, S. Song, and L. J. Guibas (2019). "Normalized object coordinate space for category-level 6D object pose and size estimation". In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Wu, J., C. Zhang, X. Zhang, Z. Zhang, W. T. Freeman, and J. B. Tenenbaum (2018). "Learning shape priors for single-view 3D completion and reconstruction". In: *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 646–662.

Wu, Z., S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao (2015). "3D shapenets: A deep representation for volumetric shapes". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1912–1920.

Xiang, Y., T. Schmidt, V. Narayanan, and D. Fox (2018). "PoseCNN: A convolutional neural network for 6D object pose estimation in cluttered scenes". In: *Robotics: Science and Systems (RSS)*.

Xie, Z., G. Berseth, P. Clary, J. Hurst, and M. van de Panne (2018). "Feedback control for Cassie with deep reinforcement learning". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1241–1246.

Xie, Z., P. Clary, J. Dao, P. Morais, J. Hurst, and M. van de Panne (2019). "Learning locomotion skills for Cassie: Iterative design and sim-to-real". In: *Conference on Robot Learning (CORL)*.

Yang, C., K. Yuan, W. Merkt, T. Komura, S. Vijayakumar, and Z. Li (2018). “Learning whole-body motor skills for humanoids”. In: *IEEE-RAS 18th International Conference on Humanoid Robots (Humanoids)*, pp. 270–276.

Yu, W., G. Turk, and C. K. Liu (2018). “Learning symmetric and low-energy locomotion”. In: *ACM Transactions on Graphics (TOG)*, p. 144.

Yuille, A. L. and N. M. Grzywacz (1989). “A Mathematical analysis of the motion coherence theory”. In: *International Journal of Computer Vision (IJCV)*, pp. 155–175.

Zeng, Y., C. Wang, Y. Wang, X. Gu, D. Samaras, and N. Paragios (2010). “Dense non-rigid surface registration using high-order graph matching”. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 382–389.

Zhang, Y., S. Song, E. Yumer, M. Savva, J.-Y. Lee, H. Jin, and T. Funkhouser (2017). “Physically-based rendering for indoor scene understanding using convolutional neural networks”. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Zia, M. Z., M. Stark, B. Schiele, and K. Schindler (2013). “Detailed 3D representations for object recognition and modeling”. In: *IEEE transactions on pattern analysis and machine intelligence*, pp. 2608–2623.

Zou, G., J. Hua, and O. Muzik (2007). “Non-rigid surface registration using spherical thin-plate splines”. In: *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pp. 367–374.