# Bimanual Interaction with Clothes

## Topology, Geometry, and Policy Representations in Robots

**Lukas Twardon**

A thesis submitted for the degree of
Doctor of Engineering (Dr.-Ing.)

Faculty of Technology
Bielefeld University

**Reviewers**

Prof. Dr. Helge Ritter
Prof. Dr. Franz Kummert

2019

Printed on permanent paper as per ISO 9706

# Declaration of Authorship

In accordance with §8(1)g of the General Doctoral Regulations of Bielefeld University, I hereby declare that

- I am familiar with the current Doctoral Regulations of the Faculty of Technology.
- I have written the thesis by myself, no parts of the text have been copied from a third party or my own assessed work without due attribution, and aids and sources used have been duly referenced and attributed in the text.
- no third party has received monetary benefits or benefits in kind, directly or indirectly, for any intermediary services or work related to the contents of the submitted thesis.
- I have not yet submitted the thesis as an assessed work for a public or other academic examination.
- I have not submitted the same paper, a similar paper, or another paper as a thesis to another university.

Place, Date

Lukas Twardon

# Abstract

If anthropomorphic robots are to assist people with activities of daily living, they must be able to handle all kinds of everyday objects, including highly deformable ones such as garments. The present thesis begins with a detailed problem analysis of robotic interaction with and perception of clothes. We show that handling items of clothing is very challenging due to their complex dynamics and the vast number of degrees of freedom. As a result of our analysis, we obtain a topological, geometric, and functional description of garments that supports the development of reduced object and task representations. One of the key findings is that the boundary components, which typically correspond with the openings, characterize garments well, both in terms of their topology and their inherent purpose, namely dressing. We present a polygon-based and an interactive method for identifying boundary components using RGB-D vision with application to grasping. Moreover, we propose Active Boundary Component Models (ABCMs), a constraint-based framework for tracking garment openings with point clouds. It is often difficult to maintain an accurate representation of the objects involved in contact-rich interaction tasks such as dressing assistance. Therefore, our policy optimization approach to putting a knit cap on a styrofoam head avoids modeling the details of the garment and its deformations. The experimental results suggest that a heuristic performance measure that takes into account the amount of contact established between the two objects is suitable for the task.

# Acknowledgments

I would like to express my gratitude to the many people who have helped me on the path toward this thesis. I am particularly thankful for the opportunity to have had Professor Helge Ritter as a supervisor. He supported me when it mattered most and inspired me to become an independent researcher. I also want to thank the other members of the dissertation committee, Professor Franz Kummert, Professor Stefan Kopp, and Thies Pfeiffer. Without the persistent help of Guillaume Walck, Robert Haschke, and the other colleagues from the AGNI Grasp Lab this thesis would not have been possible. I have also greatly benefited from Oliver Lieske's technical support. Special thanks to everyone from the lunch and coffee break group with whom I shared moments of laughter and joy. I would particularly like to thank Andrea Finke and Hendrik Koesling who were my master's thesis supervisors and introduced me to the Neuroinformatics Group. Discussions with my office colleagues have been illuminating. I greatly appreciate the valuable feedback given by Sascha Fleer. I am grateful for the financial, academic, and administrative support I received from the CITEC Graduate School. In particular, I want to thank Claudia Muhl for always listening to my questions and concerns. I am grateful to my parents, brother, sisters, and friends who have supported me along the way. Special thanks also to my nephew Mika for (his parents') permission to use the inspiring video of him learning to put on a knit cap. Last but by no means least, I would like to thank my wife Janina for her understanding, love, and support during the past few years, and my sons Niklas and Lenny for providing the motivation necessary for me to complete this thesis.

# Contents

# List of Figures

# List of Tables



# Listings

# 1 Introduction

The 1989 science fiction movie *Back to the Future Part II* paints a vivid picture of the near future (more precisely, the year 2015) to which the main character Marty McFly travels using a time machine. The film shows a number of fictional everyday technologies. For example, many people in 2015 as depicted in the movie wear smart clothing. In one scene, Marty McFly is seen putting on self-tying shoes (with so-called power laces) and a jacket with sleeves that are way too big at first but, at the push of a button, automatically adjust to the arms of the wearer. In another scene, when the main character falls into a pool and gets wet, the jacket switches to a self-drying mode. At the time of this writing, the fictional future has already been overtaken by the real present. Nevertheless, our everyday life is not dominated by such smart objects. Inspired by the movie's success, a sportswear company has recently auctioned off some self-tying shoes for several thousand dollars. Besides, a few do-it-yourself guides to making more or less usable self-adjusting jackets exist on the internet. However, it seems to be far from trivial to create smart garments that are affordable and practical for daily use.

The ideas of the present thesis are similar to those in the film in that the goal is to develop technologies that help simplify the handling of everyday objects. This is opposed to technologies for use in industry. However, in another respect, our ideas are very different from those in the movie. In general, there are two approaches to automation in the context of everyday objects. On the one hand, objects can be designed in a way that allows machines to use them or even be integrated into the objects' material. We will refer to such items as *technology-enabled objects*. To many people (including the makers of the film series *Back to the Future*), this seems to be the natural approach, and there is also some serious research in that direction. On the other hand, it is possible to develop machines and algorithms that are capable of handling objects the way they are (in the following referred to as *object-inspired technology*). This is the approach we pursue in this thesis, using the example of an anthropomorphic robot that will be equipped with clothes perception and manipulation skills. Our methods can be regarded as steps toward robots that assist humans with dressing or hang out wet laundry and, in doing so, tackle the issues described in the film very differently. There are

two key arguments for object-inspired technology. First, it should be everyone's own decision to use a specially designed product or not. For example, we do not want the user to be required to wear function-specific clothing. Second, in the future, our techniques could be implemented in a general purpose domestic robot, so that it is not necessary to develop a dedicated device for each and every task.
Creating object-inspired technology means we must look at the objects a robot is supposed to interact with in detail, describe their properties, and differentiate them from other objects. Throughout this thesis, it will become apparent that it is particularly challenging to suitably represent clothes in robots. For one thing, appearance features are typically not sufficient, especially if the task is not only recognition but also dexterous manipulation. For another thing, the shape of a garment can be very complex. In addition to this, the materials of most articles of clothing are highly deformable, which makes it very hard to model the dynamics during interaction with the robot hands and other objects explicitly and in detail. Therefore, we aim at representations that are on a higher level of abstraction. To this end, we describe the spatial (and spatio-temporal) properties of clothes more qualitatively using topological and simple geometric concepts. Moreover, we attempt to identify the most relevant parts and features of the objects involved in the tasks considered, and use only those for robotic acquisition of optimal action strategies (policies).

## 1.1 Research Questions

The present thesis aims to provide answers to open questions associated with robotic perception and manipulation of clothes. The technological long-term goal of research in this field is to enable robots to handle garments in all kinds of everyday situations. For our experiments, we picked out a few exemplary tasks such as recognizing and tracking the relevant features of different clothes, grasping and regrasping, putting on and hanging up a knit cap, putting something into a stocking, and sliding an object through a pant leg. Questions related to these problems include but are not limited to the following:

*How to deal with the variety and complexity of the tasks?*
: Robotic tasks in the clothing domain are diverse, but many of them involve multiple complex objects and a hierarchy of subgoals. To keep track of the differences and similarities, the many interaction possibilities with clothes should be categorized and common challenges should be identified.

*How to deal with the deformability of clothes?*
The main challenges in garment pose estimation and tracking are the huge number of degrees of freedom and the complex dynamics resulting from the high deformability of clothes. Therefore, effective algorithms have to reduce and search the problem space in sophisticated ways.

*Which sensory modalities can be used?*
Many modern robots are equipped with different cameras and sensors. Therefore, it is important to ask which modality is suitable for a given perception task and how the input data must be processed to be useful for solving problems related to handling clothes.

*What is the right level of abstraction for representing space?*
Spatial relationships can be described either geometrically, i.e., based on a concept of distance, or topologically, i.e., more qualitatively and without relying on a distance measure. On the one hand, geometric representations have the advantage that they arise more or less naturally from the quantitative measurements typical sensors provide. On the other hand, topological properties remain invariant under deformations frequently occuring during interaction with clothes.

*What is the right abstraction level for task state and policy representations?*
Not only the spatial configurations of clothes but also task states and action strategies can be represented at different levels of abstraction. For example, the state space could either be made up of all possible sensory inputs or it could be something related to a garment manipulation task. Similarly, robotic actions could be expressed directly in joint space or the policy space could be task-centric.

*When should learning be preferred to explicit modeling (and vice versa)?*
A question often asked in cognitive robotics is whether robots should learn by interacting with the environment or domain knowledge should be integrated to make accurate modeling and planning possible. In the clothing domain, there is a lot of useful prior knowledge available, but on the other hand, the variance resulting from deformations and differences between individual garments calls for learning from experience.

## 1.2 Scenario

In the following, we describe the scenario and experimental setup used throughout this thesis. Depending on the hardware and software available, the answers

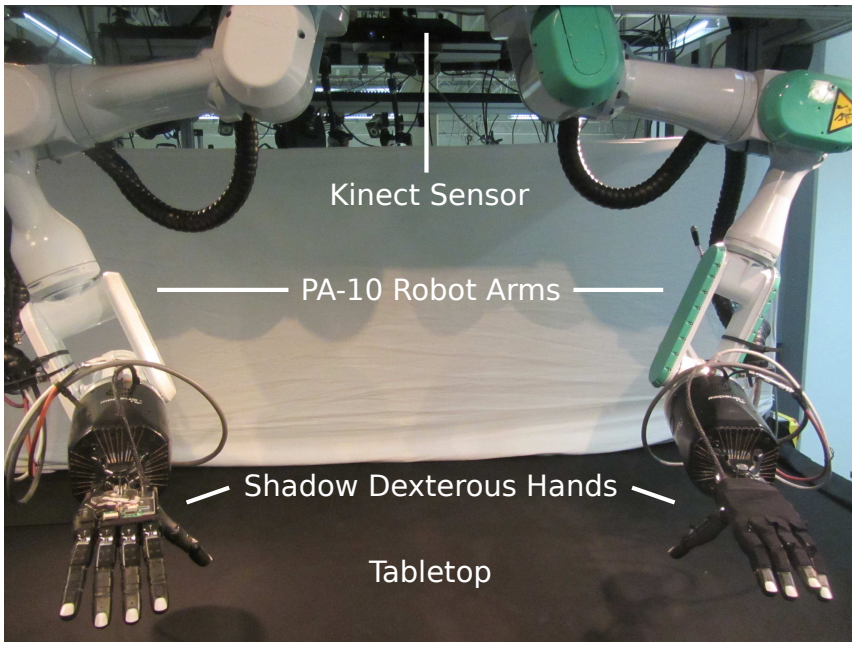



Figure 1.1: Hardware setup used in the experiments.

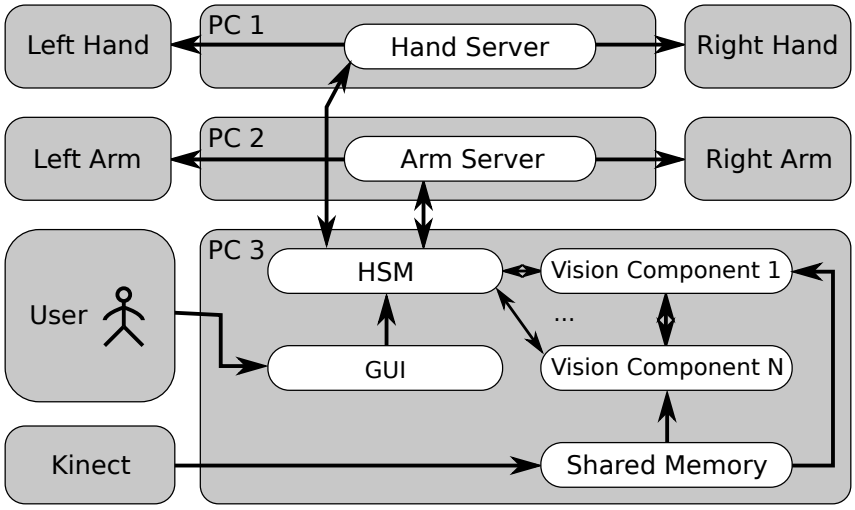



Figure 1.2: Software setup used in the experiments.

to the questions raised in Section 1.1 could be slightly different. For example, if a multi-view camera system were used, modeling geometric details would be easier because the occlusion problem would be largely avoided. Moreover, some researchers have explicitly argued for specialized robot grippers tailored to a particular task, such as the gripper for garment handling developed by Le et al. [1]. Simo-Serra et al. [2] suggested that, in general, non-anthropomorphic hand designs be used also for tasks in human environments. Nevertheless we decided to employ an anthropomorphic robot setup for the following two reasons: (i) There has been a lot of prior work in our lab and in the literature on control of and grasping with two five-fingered hands that we can draw on. (ii) Many real-world objects and tasks are designed for the human hand. Therefore, future general purpose robots are likely to be anthropomorphic.
Our hardware setup (Figure 1.1) includes two redundant 7-DOF *Mitsubishi PA-10* robot arms with attached 20-DOF electric-actuated *Shadow Dexterous Hands*. This configuration is meant to resemble the upper body of a humanoid robot. The finger tips are equipped with tactile sensors which, however, were only used for protecting the tendons in the fingers against wearout [3] and for grasp evaluation in rigid object grasping (Section 4.3.5). For a discussion about why we focus on visual perception in this thesis, the reader is referred to Section 2.1.1. For robot vision, we use a *Microsoft Kinect v1* including both a color camera and a depth sensor viewing from above toward a tabletop at an angle of about 45°.
We employ a distributed system consisting of three computers, one for each group of tasks: hand control, arm control, and vision. The inter-process communication flow is shown in Figure 1.2. Two robot server components control the hands and arms and simulate the robot kinematics in oder to avoid collisions. A shared memory provides the Kinect sensor data to the vision components. The user (or experimenter) can send commands to the robot through a graphical user interface. Overall task control is carried out by a *hierarchical state machine (HSM)*. The states of the HSM usually correspond to subtasks, and state transitions can be triggered by the user, the robot server, or a vision component.

## 1.3 Contributions and Outline

The remainder of this thesis is organized as follows: In Chapter 2, we thoroughly define the problems and tasks addressed in this work. We will be loosely guided by a well-established cognitive theory called the *problem space theory*. One result of our analysis will be the insight that naive task representations in the clothing domain are almost always highly unstructured and prohibitively

large. Other contributions include a taxonomy of interaction primitives with garments, a topological and geometric characterization of clothes, and a specification of the dressing task. Finally, we will be able to derive a few strategies for building efficiently searchable task representations. Chapter 3 is a review of related works.
Chapters 4, 5, and 6 provide reduced representations and suitable algorithms for different subproblems of the garment handling task. In Chapter 4, we present techniques for static perception of typical scenes with clothes, with application to grasping and regrasping. At the heart of this chapter are two methods for garment opening identification and a novel concept called the *boundary grasp*. In addition to this, we propose a primitive-based point cloud filter useful for data preprocessing. Chapter 5 introduces *Active Boundary Component Models (ABCMs)*, which can be seen as an extension of the concepts from Chapter 4 to the dynamic case, and we show how these models can be tracked with point clouds and applied to a simplified dressing task. Chapter 6 is concerned with policy search in a knit cap dressing scenario. We present a head-centric trajectory parameterization, a novel objective function for the task, and a suitably designed toy problem to support hyperparameter search. Moreover, the chapter makes two algorithmic contributions: (i) A method for head pose and scale estimation and (ii) a policy search algorithm combining evolutionary optimization with a simple surrogate model. In Chapter 7, we present our conclusions and give recommendations for future research.
Many of the contributions have already been published in two conference papers and a journal paper. Parts of the garment characterization presented in Chapter 2 and the interactive perception and grasping approach from Chapter 4 are covered in [4]. The formalization of the dressing task (Chapter 2), the polygon-based boundary component detection method (Chapter 4), and ABCMs (Chapter 5) are described in [5]. Besides, large parts of Chapter 6 are based on [6]. The papers and accompanying videos are available online[1][2][3].

[1] `https://pub.uni-bielefeld.de/record/2904297`
[2] `https://pub.uni-bielefeld.de/record/2908790`
[3] `https://pub.uni-bielefeld.de/record/2917731`

# 2 The Problem Space of Bimanual Interaction with Clothes

Intuitively, one might find the task of handling clothes rather simple. People accomplish it every day when they get dressed in the morning, when they do the laundry, or when they hang up their jacket after coming home in the evening. Children acquire many of the involved skills during the early stages of their development, and healthy adults are expected to master these routines effortlessly and efficiently. Nevertheless, present-day robots are still far from being able to take the work out of humans' hands. As a consequence, physically handicapped persons often have to rely on caregivers' assistance, e.g., with dressing. Apparently, our judgment of the task difficulty is biased by the importance of the task in everyday life.

To gain a better understanding of how difficult the task truly is (for a robot), we require a more formal description of the problem. To this end, we borrow a definition of problem solving from classical cognitive science. In Newell and Simon's *problem space theory* [7], solving a problem is equivalent to searching a so-called problem space. The problem space consists of an initial state, at least one goal state, and all possible intermediate states. The problem solver can choose from a range of operators, otherwise known as actions, to change the current state until a goal state is reached. According to this theory, the difficulty of the problem is determined by the size of the problem space and, more importantly, by its structure. In fact, the structure of the problem space, also called internal representation, has a strong influence on how efficiently the space can be searched.

In this chapter, we will consider both the task environment, i.e., what is more or less objectively given, and the problem space, i.e., how the task can be represented in a robot. Starting with the perceptual input and the action possibilities (Section 2.1), and continuing with the spatial properties of clothes (Section 2.2) and a specification of the dressing task (Section 2.3), we will see that the naive problem space of bimanual interaction with clothes is extremely high-dimensional and difficult to search. Therefore, we will have to find ways to reduce this space both in size and in complexity. In this connection, it will become apparent that different subproblems of the clothes perception and

manipulation problem demand different representations, and that the representation structure influences the search strategy to a great extent (Section 2.4).

## 2.1 Perception and Action in the Clothing Domain

In modern robotics, it can be assumed that the outside world (the task environment) and its internal representation (the problem space) are linked in two different ways. In one direction, information continuously flows from the environment to the robot. This link is established through sensors whose output is the input to the robot's perceptual system. In the other direction, the robot is capable of changing the environment using its actuators. Besides describing robotic perception of and interaction with clothes, we will discuss environment-contingent action possibilities (affordances), i.e., how the set of meaningful actions is determined by the properties of the environment itself.

### 2.1.1 Perception

There are three sensory modalities that could possibly play a role in bimanual interaction with clothes: proprioception, vision, and touch. In robotics, proprioception is achieved through forward kinematics, i.e., through the computation of the end-effector poses from measured joint angles. By vision, we mean the processing of a camera image, which is most commonly an RGB color image. Unfortunately, while color cameras are cheap, they do not always provide the information needed for task-relevant visual perception. To detect the configuration of an item of clothing, for instance, depth data might be much more helpful than color. Although this information can sometimes be inferred from RGB images using stereo vision or single-image depth cues, it is often easier to employ time-of-flight or structured-light sensors which output depth images directly. Regardless of the type of camera used, vision has the important advantage over other sensory modalities of providing almost global views of a scene without requiring potentially distorting physical contact.
By contrast, tactile sensors attached to the robot's end effectors (the fingers in the anthropomorphic case) provide highly localized measurements. However, in the domain of interaction with clothes, the normal forces detected by common sensors may be very low and unspecific. Consider the example of a robot holding a thin piece of cloth with two fingers such that it hangs freely under gravity. If the robot now moves the other hand toward the cloth and the fingers

begin to touch it, the fabric immediately deforms in hard-to-predict ways. At the same time, there is little or no measurable pressure against the fingertips, and if there is some, the data does not contain much information about how the configuration of the cloth has changed. Tangential force measurements could be slightly more useful because they might allow one to detect slippage or to derive shear forces in the fabric, but all inferences would be limited to local deformation effects. This is one of the reasons why, in the present thesis, we lay the focus on visual sensing and processing instead.
We point out however that, to a certain extent, the situation in our example changes when an additional object comes into play. Consider a hook located at a fixed position near the low end of the hanging cloth. If the fingers of the robot now push the cloth toward the hook, high-level or low-level force signals might be observed at the fingertips depending on whether the fabric got caught on the hook (constraining the deformation of the fabric) or not. This information is relevant in the sense that, even though gathered locally at the robot hand, it provides some knowledge of how the overall contact relation between two objects (the cloth and the hook) has changed. Detecting contact indirectly from its effects is indeed an important skill in interaction with highly deformable objects. However, our robot will almost exclusively use vision (and possibly proprioception) for that, exploiting the superior ability of visual systems to assess global and structural changes in the scene. Consider the following two examples.

*Example 1:* A robot tries to pick up a piece of clothing. From the forces measured at the fingertips, it cannot decide whether the grasp is stable or not. However, in case of success, parts of the fabric move along with the hand when lifting the arm. This effect can be perceived visually.

*Example 2:* A robot assists a person with putting on a sock. From the complex tactile and visual signals, it cannot tell whether it is successful in establishing the right amount of contact between the sock and the foot. However, if the sock slips off or does not fit properly, the effect is clearly visible.

Besides detecting spatial relations between objects, the goal of perception is very often to reconstruct the configuration of an individual task-relevant object. As a result of this process, one usually obtains the parameter values of a finite-dimensional model. The number of parameters needed to describe the configuration of an object in space depends on the type of object considered.

*Rigid objects:* Solid bodies in which deformation can be neglected are referred to as rigid objects. The pose of an unconstrained rigid object (such as a

plastic cube) can be specified by six parameters, three for describing its position and three corresponding to its orientation. This is another way of stating that the object has six degrees of freedom.

*Articulated objects:* To describe the configuration of an articulated object, i.e., an object consisting of several rigid parts connected by joints (such as a folding ruler), an extra parameter is required for each degree of freedom added by a joint.

*Locally deformable objects:* The configuration of a locally or reversibly deformable object (such as a sponge) can be represented by a rigid model together with a description of the relative deformation. This can be any parameterized mapping from an undeformed reference configuration to the deformed state of the object.

In clothing, deformations tend to be global, hardly reversible (due to the limpness of the material), and so complex that none of the mentioned approaches can be used. The huge amount of degrees of freedom that one would have to model renders exhaustive representations impractical. Moreover, the situation is often complicated further by the complex interplay between the considered garment and other physical entities such as the robot and, depending on the task, additional deformable or rigid objects. The following classical problems of visual perception are hence usually much harder to solve in interaction with clothes than in other domains.

*Segmentation:* Before the configuration of a single object or relations between different objects can be detected, the relevant entities need to be segmented from each other and from the background. In the clothing domain, this is particularly difficult because common segmentation criteria, such as sharp edges between objects, do not apply if a garment wraps around or lies flat on another object. Then, using a combination of geometric and color cues may be the only practical solution.

*Recognition:* If the type of object that the robot is supposed to interact with is not known a priori, the garment category (T-shirt, pair of pants, sock, etc.) has to be classified on the basis of visual information. However, clothing exists in all colors and designs, and with a wide variety of prints. Therefore, classifiers have to rely on geometric features, which may be difficult to extract from images of heavily deformed garments, though.

*Pose estimation under (self-)occlusion:* Not only are models of garments and their configurations usually high-dimensional, but the data on which they

are based is almost always incomplete. First, a garment may be partially hidden by another object (occlusion). Second, and more importantly, some parts of the garment may be obscured from the camera view by other parts that are closer to the camera (self-occlusion). While symmetry or continuity assumptions often facilitate model completion of rigid objects, it is nearly impossible to reconstruct the configuration of the invisible parts of a highly deformable garment. Therefore, one always has to make sure that the relevant parts are seen by the camera.

*Tracking:* To handle incompleteness, noise, and ambiguities in images of moving objects, tracking (i.e., locally updating a model over time rather than estimating the object pose from scratch at each frame) is an essential skill. On the one hand, this requires making assumptions about which configuration changes are plausible. On the other hand, it is important not to over-constrain the model, given the fact that clothes may indeed be subject to strong deformation.

### 2.1.2 Action

Despite the high dimensionality and complexity of the problem, it is possible to find suitable representations, considering the fact that a complete model of the environment is not necessarily required for accomplishing a task. We only have to define the problem space such that a task solution can be found using the actions available to the robot. With this in mind, we have created a taxonomy of interaction primitives with clothes (Table 2.1). By analogy with the twofold goal of perception (to reconstruct the configuration of individual objects and to detect spatial relations between objects), we have categorized the interaction primitives according to their effect, i.e., according to whether they only manipulate the configuration of the involved garment or change the spatial relation of the garment to another object. Moreover, it has become apparent that the openings of the garment (e.g., at the waist, leg, or sleeve ends) play a prominent role in many of the interaction primitives. Therefore, another grouping criterion has been whether garment openings are central to the interaction or not. Interestingly, most interaction primitives with clothes can be paired with an "inverse" primitive that, to some degree, reverses the effect of its counterpart. Primitives and their respective inverses have been placed side-by-side in Table 2.1.

While the vast majority of interactions preserve the number of garment openings and their global structure, *opening* and *closing* (e.g., by (un)zipping or (un)buttoning) form an exception to this rule. *Folding outward* (e.g., a collar or a sleeve) is another example of changing the configuration of a piece

Table 2.1: Taxonomy of interaction primitives with clothes

<table>
<tr><th>interaction primitive</th><th colspan="2">involves garment openings</th><th colspan="2">does not involve garment openings</th></tr>
<tr><td rowspan="3">changes the garment configuration</td><td>opening</td><td>closing</td><td>crumpling/ deforming</td><td>stretching/ flattening</td></tr>
<tr><td rowspan="2">folding outward/ turning inside out</td><td rowspan="2">folding inward/ turning outside in</td><td>folding</td><td>unfolding</td></tr>
<tr><td>twisting/ knotting</td><td>untwisting/ unknotting</td></tr>
<tr><td rowspan="2">changes the contact relation to the robot hand</td><td>grasping (targeted)</td><td>releasing</td><td rowspan="2">grasping (randomly)</td><td rowspan="2">releasing</td></tr>
<tr><td>reaching into</td><td>pulling the hand out</td></tr>
<tr><td rowspan="2">changes the contact relation to the support surface</td><td rowspan="2">-</td><td rowspan="2">-</td><td>picking up/ lifting</td><td>dropping</td></tr>
<tr><td colspan="2">sliding/pushing (across/off the surface)</td></tr>
<tr><td rowspan="4">changes the contact relation to another object</td><td>hanging up (targeted)</td><td>unhanging</td><td>hanging up (randomly)</td><td>unhanging</td></tr>
<tr><td>pulling over</td><td>pulling off</td><td rowspan="3">wrapping sth.</td><td rowspan="3">unwrapping sth.</td></tr>
<tr><td>putting sth. in</td><td>taking sth. out</td></tr>
<tr><td>sliding sth. through</td><td>pulling sth. out</td></tr>
</table>

of clothing by manipulating its openings. Carrying this interaction primitive to an extreme by pulling one of the openings (let us call it $o_1$) over all other openings ($o_2, ..., o_n$) and/or closed ends ($e_1, ..., e_m$) turns the garment inside out. This is equivalent to folding $o_2, ..., o_n$ and $e_1, ..., e_m$ *inward* and pushing them through $o_1$.

Manipulations which do not involve garment openings tend to be less complex. In principle, *crumpling, folding, twisting,* and *knotting* are all instances of deforming the surface of a piece of clothing. However, *crumpling* results in unstructured deformations, whereas *folding* and *twisting/knotting* produce structures of creases and coils, respectively.

We emphasize that, in our experiments, we focus on those interactions which change the contact relation between a garment and either the robot or another physical entity because these primitives provide the basis for complex skills in scenarios with multiple objects. In manual interaction with clothes, establishing contact between a garment and the hand of the robot is of course essential. We distinguish between two types of *grasping*. If the sole purpose of grasping is to attach the garment to the fingers, the contact position is arbitrary as long as it allows a stable grasp. We refer to this as *random grasping*. By contrast, if the grasp serves to make another action possible (e.g., to pull the garment over another object), it usually has to be *targeted* (e.g., around the boundary of an opening). A further primitive that possibly establishes contact between the garment and the robot hand is *reaching into* one of the openings.

In some of the interactions, a support surface, such as the floor or a tabletop, plays an important role. Obviously, *picking up* or *dropping* an item of clothing changes the contact relation to the support surface. *Sliding the garment across* the surface only changes the contact position, whereas *pushing it off* the surface (e.g., the tabletop) is another way of changing the contact relation globally.

Concerning the interaction primitive of *hanging up* a garment, we again distinguish between a *random* and a *targeted* version. In some cases, it is sufficient to hang the garment over the target object (e.g., a coat rack or hook) in an arbitrary manner, and to rely on gravity and the deformability of the fabric that cause it to get caught. In other cases, it is necessary to make use of an opening (e.g., of a hood) in order to be successful.

*Wrapping* is often the only way to create a stable attachment of an arbitrary object to a garment without openings. If the garment has at least one opening, depending on its orientation in space, it may be possible to make use of gravity to *put something in*. Using the deformability of a garment to *pull it tightly over* another object is probably among the most common interaction primitives (e.g., when putting on a sock). We note that, in our analysis of interaction primitives with clothes, we make no distinction between a human body part

and any other physical object. In fact, *sliding a limb through* a piece of clothing is one of the most important interactions during the task of getting dressed or undressed. This primitive requires the garment to have at least two connected openings.
To give an example of how everyday tasks are composed of several instances of the listed interaction primitives, we consider the following scenario:
*Alice* enters her apartment and is about to take off her winter clothes. She performs the actions below (using the interaction primitives given in brackets).

1. She puts the keys back in her pants pocket *[putting sth. in]*.
2. She grasps her knit cap *[grasping (randomly)]* and pulls it off her head *[pulling off]*.
3. She regrasps the cap around the boundary of its opening *[grasping (targeted)]* and hangs it up on the coat rack *[hanging up (targeted)] [releasing]*.
4. She unzips her jacket *[opening]*.
5. She reaches into the left sleeve of her jacket with two fingers of the right hand *[reaching into]*, grasps it around the boundary *[grasping (targeted)]*, and pulls her left arm out *[pulling sth. out] [releasing]*.
6. She reaches into the right sleeve *[reaching into]*, grasps it *[grasping (targeted)]*, and pulls the right arm out *[pulling sth. out]*.
7. She regrasps the jacket around the boundary of the hood *[grasping (targeted)]* and hangs it up *[hanging up (targeted)] [releasing]*.
8. She grasps her scarf *[grasping (randomly)]*, takes it off *[unwrapping sth.]*, and throws it on the coat rack *[hanging up (randomly)] [releasing]*.
9. She takes off her gloves *[grasping (randomly)] [pulling off]*, crumples them in her hands *[crumpling]*, and puts them in the laundry basket *[dropping]*.

### 2.1.3 Affordances

It seems natural to integrate a database of interaction primitives into the memory of a robot as part of its prior knowledge about clothes, possibly influencing in a top-down manner how the robot perceives the environment. However, according to [8], there is also a bottom-up aspect of perception, i.e.,

the environment itself has an effect on the structure of its representation and on the action possibilities available to the robot. In this connection, the term *affordance* plays an important part.

> The affordances of the environment are what it offers the animal, what it provides or furnishes, either for good or ill. ... [The term] implies the complementarity of the animal and the environment.
>
> — Gibson [8, p. 127]

Of course, in our case, *animal* is to be replaced by *robot*. The meaning of affordances is best explained using examples: The shape of a zipper provides an affordance for *opening* or *closing*. The deformability of most clothes affords *crumpling* or *folding*. A pants pocket, together with a small rigid object, offers the robot an affordance for *putting the object in* the pocket. In a similar manner, we could express all interaction primitives with clothes as affordances arising from the physical properties of the involved objects.
As already pointed out by Gibson in his original work, detecting affordances is not necessarily the same as classifying an object. A pocket affords putting something in, regardless of whether it is a pants pocket or the breast pocket of a shirt. Nevertheless, affordance sometimes implies classification and vice versa: If something affords sliding the legs through, it is probably a pair of pants. Conversely, if something has been classified correctly as a sock, it inevitably provides an affordance for pulling it over a foot. However, just because an object affords a particular interaction does not mean the robot notices it. Affordances themselves are independent of perception [9]. Consequently, there may be false positives (*false affordances*) as well as false negatives (*hidden affordances*). For example, a printed breast pocket may create the illusion that something can be put in, and the real pocket of a crumpled shirt may be hidden by other parts of the garment.

## 2.2 The Topology and Geometry of Clothes

Rather than directly addressing the problem of how robots can *detect* affordances, we first approach the question of which properties cause clothes to *provide* certain affordances. To this end, we describe the spatial structure of garments more or less objectively and globally, i.e., we take the perspective of an omniscient observer (as opposed to the limited perspective of the robot's perceptual system). By doing so, we avoid being deceived by the misleading local features of false affordances. In the following, we also assume that

Figure 2.1: Distances in 3-space between points on the surface of objects before and after typical manipulations. (a) On a rigid object, the distance between two arbitrary points is preserved because translation and rotation are the only possible transformations. (b) On a deformable garment, the distances may change dramatically when the object is manipulated, e.g., by a robot.

color and the reflective properties of the material in general do not have any influence on the affordances clothes provide.
Although the goal of this analysis is a global decription of clothes, it is necessary to first consider the basic structures that make up garments: Most items of clothing are made of textiles, i.e., networks of yarn which are created, e.g., by weaving or knitting. One natural way to regard clothes is hence as meshes or graphs consisting of vertices representing the intersection points between the strands of yarn, and edges connecting adjacent vertices. From now on, we will refer to such representations as *textile graphs*. However, the fabric of most garments is so finely woven that we can introduce some abstraction and consider the clothing material as forming a surface, which is an essential concept both in topology and geometry.

### 2.2.1 Topology

While Euclidean geometry is concerned with numerical measurements such as distances between points, angles between lines, as well as the shape, size, and position of objects, topology [10] provides more qualitative descriptions of space. When clothes are deformed in typical ways, many of their quantitative properties change continuously over time. In particular, and in contrast to rigid objects, the distances in Euclidean 3-space between different points on the garment surface are usually not preserved (Figure 2.1). But there are also object properties that remain invariant under continuous deformation. Since topology is the study of just these properties, it is well-suited for describing the spatial structure of clothes in a very general way.
One of the basic concepts of topology is that of *topological spaces*, which can be defined without any notion of distance. The most common definition of a

topological space is as a set $X$ of points together with a collection $T$ of subsets of $X$. The subsets are called open sets and the collection $T$ is called a topology on $X$ if the following axioms are satisfied:

1. The empty set and $X$ belong to $T$.
2. The union of an arbitrary number of members of $T$ belongs to $T$.
3. The intersection of a finite number of members of $T$ belongs to $T$.

Any metric space has a topology that is said to be *induced* by its metric (distance function). Given a metric $m$, the topology can be constructed from a base formed by the open balls $B_r(x_0) = \{x \in X : m(x_0, x) < r\}$, where $x_0 \in X$ and $r > 0$. The collection of open sets then consists of all sets which are unions of open balls. For example, a hypothetical textile graph can be considered as a *simplicial 1-complex*[1] in which the edges of the graph are copies of the real unit interval $[0, 1]$ glued together at the vertices. This complex is naturally equipped with a metric indicating the shortest path between two given points (which can be vertices of the graph or points on an edge). Then, the open ball $B_1(v_0)$ centered at a vertex $v_0$, for instance, contains $v_0$ and all its adjacent edges. The topology induced by this metric consists of all possible unions of all such open balls centered at any of the points in the complex.
In the following, we focus on surfaces, but we note that we can think of a textile graph being *embedded* on a surface, and thereby inheriting many of its properties [11]. In particular, homeomorphism and connectedness have equivalent meanings in graphs and surfaces, and the genus of a graph denotes the minimal integer $g$ such that the graph can be drawn on an oriented surface of genus $g$ without self-crossings. The terms homeomorphism, connectedness, genus, and orientability are defined and explained below.
Surfaces in the topological sense are two-dimensional manifolds. An *n-manifold* is a locally Euclidean topological space, i.e., a space in which each point has a neighborhood (a subset of the space that includes an open set containing the point) that is homeomorphic to the Euclidean space of dimension $n$. Two spaces are called homeomorphic if a homeomorphism exists between them. *Homeomorphism* is another fundamental concept of topology. It is defined as a function $f$ between two topological spaces which has the following properties:

[1] *Simplices* are the simplest forms of polytopes (generalized polygons) consisting of only $n + 1$ vertices, where $n$ denotes the dimension of the simplex. For example, a 0-simplex is a point, a 1-simplex is a line segment, a 2-simplex is a triangle, a 3-simplex is a tetrahedron, and so on. A *simplicial n-complex* is a set $S$ of simplices of dimension $n$ or lower that satisfies the following conditions: (i) Any face (the convex hull of any nonempty subset of the points that define the simplex) of a simplex $s \in S$ is also in $S$. (ii) The intersection of any two simplices $s_1, s_2 \in S$ is empty or a face of both $s_1$ and $s_2$.

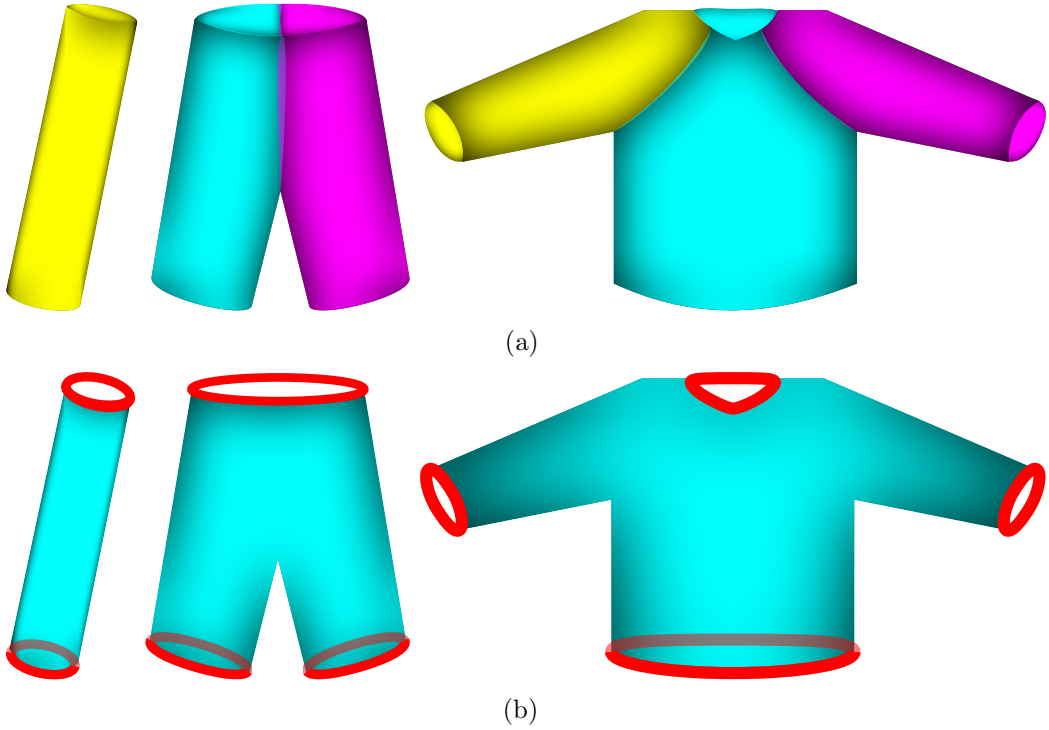
1. $f$ is a bijection.
2. $f$ is continuous.
3. $f^{-1}$ is continuous.

Formally, a function $f : X \to Y$ is continuous if the inverse image of any open set in $Y$ is also an open set in $X$. But the concept of continuity is well explained informally using the example of manipulating clothes. Considering the surface of a piece of clothing as a 2-manifold, the properties that make a transformation applied to the garment a self-homeomorphism roughly have the following meaning:

*Continuity of $f$:* Intuitively, a garment manipulation is continuous if it does not involve cutting, ripping, or any similar action. In particular, most interaction primitives from Section 2.1.2 are continuous, with the exception of *opening* (by unzipping or unbuttoning).

*Continuity of $f^{-1}$:* The inverse function of a transformation is continuous if no gluing or, more commonly in the domain of clothing, sewing, zipping, or buttoning is required. As this is also true for most interaction primitives with clothes (except for *closing*), they can be considered as self-homeomorphisms of the involved garments.

One way to think of clothes is as 2-manifolds without boundary comprising both the outer and the inner surface of a garment (Figure 2.2(a)). A piece of clothing specified in this way has some topological invariants, i.e., spatial properties that do not change under typical continuous deformations (self-homeomorphisms). One such invariant is *connectedness*, which is defined as follows: A topological space is connected if it cannot be written as the union of two disjoint nonempty open sets. Usually, the fact that a garment can be regarded as a connected space makes it an identifiable object and separates it from other objects. However, there are exceptions to this rule. For example, a pair of zip-off pants, i.e., a pair of pants with removable legs, may, in some sense, be considered a single garment even if the legs have been removed.
The *genus* of a connected (and oriented) surface is an invariant that plays an important role in the topological characterization of clothes. It is defined as the maximum number of cuttings along non-intersecting and non-self-intersecting closed curves (loops) that can be performed without disconnecting the surface. In an intuitive sense, it is the number of holes in the surface. Thus, scarfs, caps, gloves, socks, and other similar garments are represented by genus zero



(a)

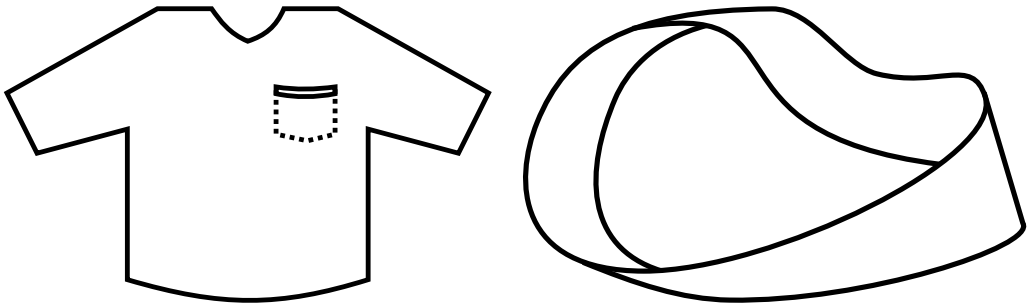
(b)

Figure 2.2: Two ways of regarding the topology of garments. (a) A leg warmer, a pair of pants, and a sweater represented by 2-manifolds without boundary. The garment surfaces are thought of as connected sums of $g$ stretched tori (sleeves). (b) The outer surfaces of the same items of clothing considered as genus zero 2-manifolds with boundary. The boundary components are depicted in red.

manifolds. Closed surfaces of genus $g \geq 1$ with properties as above can be expressed as connected sums of $g$ tori. In this regard, a torus can be visualized as a manifold homeomorphic to the surface of a doughnut, and forming the connected sum of two tori means removing a disc from each torus and gluing them together along the resulting boundary circles. In clothing, stretched torus-like structures often occur in the form of sleeves. Hence, one can naturally consider garments as connected sums of sleeves, i.e., as sleeves suitably sewed together. In fact, the garments shown in Figure 2.2(a) can be classified by their genus, i.e., by the number of sleeves they are made up of: A leg warmer has genus one, a pair of pants has genus two, and a sweater has genus three.

A slightly different way to think of clothes is as genus zero 2-manifolds with boundary (Figure 2.2(b)). From this perspective, the fabric has no thickness, and only the outer surface of a garment is regarded. Then, the essential topological invariant is no longer the genus, but the number of *boundary components*. In this context, the boundary is the complement of the manifold's interior (the set of points with neighborhoods homeomorphic to $\mathbb{R}^n$), and the boundary components are the connected components of the boundary. The boundary components themselves are manifolds without boundary of dimension $n - 1$, i.e., in our case, they are closed curves (1-manifolds). Apart from garments without openings such as scarfs, the boundary components usually coincide with the boundaries of the garment openings., i.e., caps, gloves, or socks have one, whereas leg warmers, pairs of pants, and sweaters have two, three, and four boundary components, respectively.

We note that some special structures of clothes are impossible to represent as oriented 2-manifolds with boundary in the same way as above. For example, a breast pocket, considered as a surface sewed on another surface, is non-manifold because it has edges with three neighboring surfaces and hence does not locally resemble Euclidean 2-space in a neighborhood of every point (Figure 2.3(a)). Furthermore, we have specified that a manifold with boundary should represent a garment's outer surface, which is implicitly assumed to have an unmodeled complementary inner surface. In other words, the surface is required to be *orientable*, i.e., it must be possible to consistently define it as either the inner side or the outer side of a garment. However, there are some rare examples of clothes in which this is impossible: A loop scarf that has been created by twisting a regular scarf 180 degrees and joining the ends resembles the so-called Möbius strip and is hence not orientable (Figure 2.3(b)). Nevertheless, on a certain level of abstraction and if we make a few assumptions, the topology of most garments is uniquely characterized by the number of boundary components.



(a) (b)

Figure 2.3: Garment structures that cannot be represented by oriented 2-manifolds with boundary. (a) A breast pocket regarded as a surface with boundary sewed on the surface of a shirt. The resulting space is non-manifold because the edges depicted as dashed lines have three neighboring surfaces which violates the axiom that a 2-manifold should be locally homeomorphic to Euclidean 2-space. (b) A loop scarf shaped like a Möbius strip has only one non-orientable surface which cannot be defined as the garment's outer side in a consistent manner.

### 2.2.2 Geometry

The topological invariants provide a very general and global description of a garment's spatial structure because they are preserved under the most common manipulations performed, e.g., by a robot. However, the informative content of these invariants is also limited in that they do not tell us anything about the exact shape or pose of the garment. For example, the statement that the surface of an item of clothing is *connected* translates as "It consists of one piece.", which naturally raises the question, "A piece of what?". Topology alone cannot answer this question since very different objects may be topologically the same (e.g., a sock and a glove are represented by homeomorphic manifolds).

Throughout our topological analysis of clothes, we have been interested in metrics only in so far as they helped us to define reasonable topologies on garments. For example, in a textile graph, the geometric realizations of the edges could, from a topological perspective, be arbitrary real intervals because they are all equivalent up to homeomorphism. However, in order to accurately model the garment's geometry, they should reflect the true physical lengths of the strands of yarn. Of course, it may be impractical to keep track of such details, but similar considerations hold for the more abstract representation

by manifolds. Although the distances between points on a garment are not preserved in 3-space, the shortest-path distances on the surface remain almost constant under a certain class of deformations. In this context, we distinguish between stretching, which may significantly change the distances on the manifold, and bending, under which they are preserved to the greatest extent. Depending on the properties of the material, we can thus make some geometric assumptions. For example, if the fabric of a sweater is not too elastic (stretchable), the length of the sleeves may be assumed to be constant.
It is however difficult to make general statements about the geometry of clothes because there may be great differences between items of a particular garment category (e.g., between different types of pants). Moreover, the embeddings of the garment surfaces in 3-space tend to be complex. One way to circumvent this issue is to assume a canonical configuration of and a certain perspective on a garment. For example, a short-sleeved shirt spread out on a flat surface and viewed from above has a T-shape, and a pair of pants roughly has a Y-shape. Rather than considering the overall geometry of an item of clothing, it is also possible to focus on the most important parts (such as the closed curves representing the boundary components). This outlines one of the key challenges in interaction with clothes, which is to find task-specific geometric heuristics that can be used in addition to the general topological knowledge we have about clothes.

### 2.2.3 Effects on Perception and Action

The topological and geometric properties of clothes per se are part of the task environment and as such independent of perception. Conversely, they do however influence the representation of the task in the robot's perceptual system in two different ways. First, the topology and geometry of garments imply affordances that may or may not be perceptible from the sensory input alone, i.e., in a bottom-up manner. Second, the robot's topological and geometric prior knowledge of clothes leads to certain expectations and hypotheses that guide perception (top-down processing) and thus robotic action.
The characteristics of garments that induce affordances can be categorized into topological, geometric, and material properties. We give a simple example of a topology-induced affordance: A pair of zip-off pants consisting of three detached parts (two legs and the top part) provides an affordance for connecting the parts by zipping the legs back on (and thus creating a connected manifold). Some of the most important affordances of clothes are effects of the number of boundary components: If a garment has at least one boundary component that represents an opening, it can be used, e.g., for pulling it over another

object. If it has two or more boundary components, it potentially affords sliding an object through. But it is often a geometric property that makes an action physically possible or impossible. The affordance of sliding a cylindrical object through the opening of a garment, for instance, becomes realizable only if the circumference of the cylinder base is smaller than the arc length of the boundary component corresponding to the opening. We note however that the material properties of the garment might provide an affordance for changing the geometry, e.g., by stretching it such that the object fits through the opening.
During the perception of clothes, robots are confronted with complex and ambiguous stimuli. Therefore, it is often difficult to detect affordances directly from the sensory input. Instead, it may be possible to use abstract topological or geometric prior knowledge to initiate a top-down process. For example, if the robot knows that the only object in its field of view is a garment whose surface is connected, it can exploit this knowledge and apply a simple foreground-background segmentation algorithm, rather than trying to solve the much harder problem of segmenting an arbitrary scene. Similarly, if the garment category is known, detecting openings simplifies to finding a predefined number of boundary components.

## 2.3 The Dressing Task

In Section 2.1, we have described the perception of and interaction possibilities with clothes without taking into account their overall spatial structure. By contrast, in Section 2.2, we have focused on the form of garments but largely ignored the purpose it serves. However, what all garments have in common is their main function which is to cover parts of the body. Therefore, one reason to analyze the dressing task in detail is the fact that robotic assistance with dressing could be helpful, especially for physically handicapped persons. But beyond that, we also believe that, since dressing is the inherent purpose of clothes (this is what they were made and designed for), the task determines the relationship between the form and function of garments. As a consequence of this analysis, we therefore hope to gain some insight into how the objects involved in the task and the interactions between them could be represented in robots, not only during dressing assistance but also during other manipulation tasks in the clothing domain.
We first formalize the dressing task topologically, i.e., without considering the exact geometry of the involved objects (the human body and one or more items of clothing). To this end, we draw on the spatial characterization of clothes from Section 2.2 along with a simple topological representation of the

body and relate them through qualitative dressing paths. While this simplistic description of the task neglects all difficulties that arise when performing it in the real world, we also decompose the task into smaller subproblems that make it much easier to identify the challenges of robot-assisted dressing.

### 2.3.1 Task Topology

In the following, we describe the task of putting on a piece of clothing in a topological manner. We assume that the outer surface of the garment is represented by a connected, orientable genus zero 2-manifold with boundary. Then, the garment is topologically characterized by the number of boundary components. We note that, in principle, this is also true for clothes like button shirts where, in the unbuttoned state, the collar forms a large boundary component together with the front and the waist part. However, in such cases, the dressing task usually involves an opening (unbuttoning) step before and a closing (buttoning) step after the actual procedure of putting the garment on. This means that the garment topology changes two times during dressing. In the task description below, we limit ourselves to topology-preserving strategies, i.e., in the example of button shirts, we assume them to be closed throughout the whole dressing procedure.

A very intuitive representation of the human body is that of a stick figure, which we will refer to as a skeleton. There is a graph-theoretic realization of such a skeleton (as a tree, i.e., as a connected graph without cycles). Hence, topologically, we can formalize skeletons as simplicial 1-complexes, much like we did with textile graphs in Section 2.2.1. This representation has the advantage over more realistic models in that it abstracts the big picture from geometric details such as body proportions and size. Essentially, it reduces the shape of the body to a structure of links between its extremities (the head, the hands, and the feet), which renders the skeleton models of all (non-amputated) human bodies topologically equivalent, allowing us to make some person-independent statements about the task.

In most cases, only a part of the body is involved in putting on a garment, so we usually consider a sub-skeleton. To formalize the tasks of putting on a leg warmer, a pair of pants, and a sweater, we use the skeletons representing one of the legs, the lower body, and the upper body, respectively (depicted as orange lines in Figure 2.4). If, for simplicity, we assume star-shaped sub-skeletons with one central point (depicted in green), it is easy to see that the three simplicial 1-complexes considered are pairwise non-homeomorphic because removing the central point yields two (in the single leg case), three (in

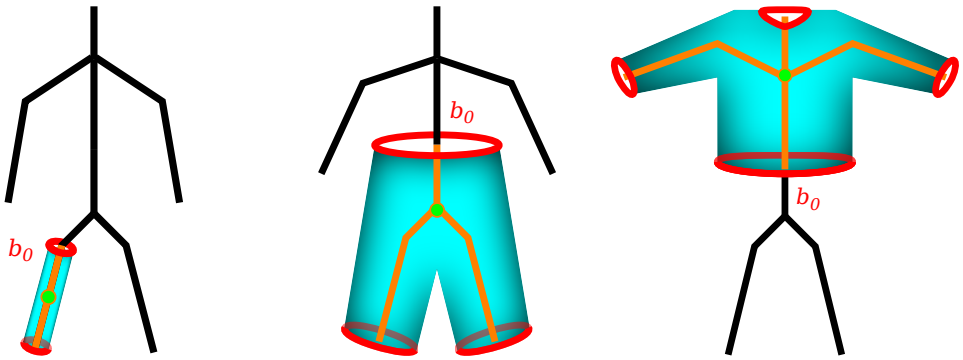



Figure 2.4: Skeleton model of the human body and target configurations of the dressing task, exemplified by a leg warmer, a pair of pants, and a sweater. The involved sub-skeletons are depicted in orange. The green dots show the central points of the star-shaped sub-skeletons.

the lower body case), or four connected components (in the upper body case).[2] Thus, the body parts involved in dressing are topologically characterized by the number of sub-skeleton end points.

To specify the objective of the dressing task, it is helpful to imagine a target skeleton in the interior of the garment which connects the centers of the boundary components (openings). We note that there are also articles of clothing that are supposed to cover one or more of the extremities (e.g., a hat covers the head and tights cover the feet). Then, the target skeleton can be considered to have end points representing the closed ends of the garment. In any case, the goal of putting on a piece of clothing is to match the relevant sub-skeleton of the body with the visualized target skeleton inside the garment. In particular, it is important that the end points of the two skeletons be positioned in a defined way with respect to each other. For example, after putting on a sweater, the end point representing the left hand should match the end point at the left sleeve opening, the head should match the neck opening, and so on.

The actual dressing procedure begins with the identification of a boundary component $b_0$ that corresponds to a specific opening through which the whole body part to be dressed has to pass. In fact, the choice of $b_0$ is uniquely predetermined by the garment category (the waist end in case of a sweater or a

[2] It can be shown that if $f : X \to Y$ is a homeomorphism, then for any $x \in X$, the restricted function $f|_{X\setminus\{x\}} : X \setminus \{x\} \to Y \setminus \{f(x)\}$ is also a homeomorphism. If we choose $x$ to be the central point of a star-shaped skeleton with $n$ branches, we obtain $n$ connected components, whereas removing an arbitrary point from a skeleton with $m < n$ branches yields at most $m < n$ connected components. However, two spaces with different connectedness properties cannot be homeomorphic because, intuitively, cutting or gluing would be necessary.

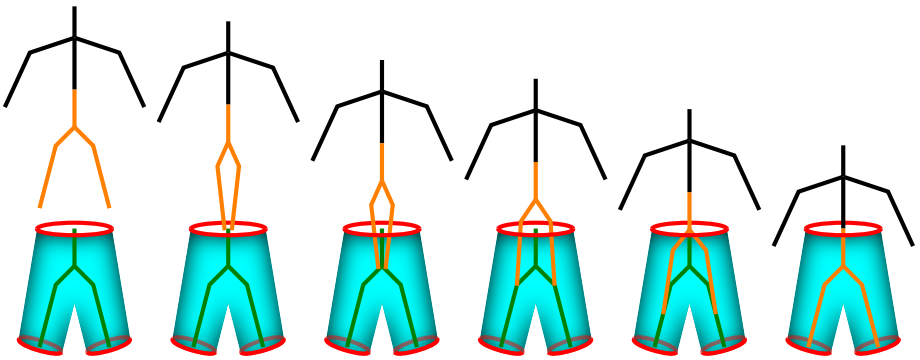


Figure 2.5: Task topology of putting on a pair of pants. The target skeleton and the sub-skeleton representing the lower body are depicted in green and orange, respectively.

pair of pants, the upper end in case of a leg warmer, etc.). Suitable trajectories through the garment can be found by considering the configuration of the target skeleton. Except for some unusual dressing strategies which involve turning the item of clothing inside out, the path of any end point of the body's sub-skeleton through the garment interior starts at $b_0$. An exemplary path then follows the target skeleton, passes its central point, and ends after passing through the associated target opening (or when reaching the associated closed end).

Figure 2.5 shows a possible trajectory of the skeleton during the task of putting on a pair of pants. We emphasize however that this is only a qualitative illustration of the procedure which, in the real world, is subject to different variations. First, the trajectory shown describes *relative* changes in pose between the skeletons, i.e., in order to achieve a certain configuration, either the human limbs or the garment could be moved. Second, we have considered the paths of the skeleton end points independently of each other. In the illustrated example, the feet move simultaneously, but they could also pass through the garment interior one after the other. Third, the garment (and thus the target skeleton and the trajectories) may be subject to topology-preserving deformations such as stretching or bending.

### 2.3.2 Task Decomposition

Up to this point, we have only considered the qualitative spatio-temporal structure of the dressing task. In doing so, we have consistently ignored the mechanical processes that are responsible for moving the garment and the body with respect to each other. For the topological analysis of the task, it has

also been irrelevant whether a person gets dressed with or without help from a robot. By contrast, in the following, we focus on robot-assisted dressing. We decompose the problem into three essential skills a bimanual robot needs in order to be able to accomplish the overall task.

*Generate a suitable initial configuration:* Even before the actual dressing procedure, the robot has to solve the difficult problem of bringing the involved entities (the garment, the human body, and the robot's own mechanical components) into a starting pose that facilitates the planning of successful dressing paths. In this regard, the garment opening that is represented by boundary component $b_0$ almost always plays an important part. First, the opening has to be found. Depending on the garment category and configuration, this may require physical actions that make the opening visible. Then, the garment has to be manipulated such that the area of the opening is increased to make the limbs fit through. Furthermore, it may be the case that the person to be dressed requires support with moving the extremities of the body to the garment opening. Besides, in order to be able to perform the subsequent steps, the robot hands must be attached to the garment, which, depending on the dressing strategy, involves one- or two-handed grasps.

*Assist with sliding the limbs through the garment:* One possible realization of an abstract relative dressing trajectory is to keep the garment in a more or less static configuration and to slide the limbs through the garment interior. Depending on the physical abilities of the person to be dressed, robotic assistance may be necessary, or the movement has to be carried out completely by the robot. This means that the robot hand grasps around one of the human limbs, and the robotic joints are controlled in such a way that the extremity of the limb is guided along the planned trajectory to its target position. As a consequence of attaching the robot hand to the person's body, the human and the robotic kinematics have to be considered as a whole. Moreover, one has to keep in mind that, in this strategy, only one robot hand is available for holding the garment.

*Pull the garment over the limbs:* Alternatively or in addition, the item of clothing can be moved and deformed while assuming the body to be static. An optimal dressing assistance robot would probably apply both strategies alternatingly. Pulling the garment over the body has the advantage that the robot can use both hands to grasp the garment. Using this strategy, the robot is faced with two competing challenges. On the one hand, the fabric can get caught on an extremity (e.g., the foot when putting on

a pair of pants) or on another body part (e.g., the knee). This may be detectable indirectly at the robot hand as the garment starts to slip out between the robotic fingers, and the robot could try to avoid such situations. On the other hand, it has to be ensured that enough contact is established between the garment and the human body. In order to pull a garment tightly over a limb, the robot must apply higher forces at some points to counteract the resistance of the body. In many cases, the exact trajectory and/or the optimal force profile may only be determined by trial and error, which poses the additional challenge of guaranteeing the safety of the person to be dressed and the involved objects during the learning phase.

### 2.3.3 Implications for Object and Task Representations

The action possibilities determine the design and appearance of most everyday objects, a fact that is often paraphrased as "form follows function". Therefore, the purpose of an object should be considered when defining the structure of its representation. From the formalization of the dressing task, i.e., from the inherent function of garments, we now derive a few guidelines for how clothes and their interplay with other objects can be modeled in robots. We think that the memory of any robot that is meant to interact with clothes in complex ways requires at least some of the following components.

*Overall plan with subgoals:* Using the example of robot-assisted dressing, we have seen that, although the movements and manipulations themselves tend to be continuous, complex interactions with clothes can be decomposed into smaller discrete subtasks with their own goals and challenges. An explicit overall plan should therefore define the conditions for transitioning from one phase to another. Moreover, the limited resources should be taken into account. For example, the plan should specify how a bimanual robot can use both hands in an optimal way. One possible implementation of such a plan is as a hierarchical state machine (Section 1.2).

*Representation of the garment openings:* It has become apparent that the openings characterize an item of clothing surprisingly well, not only from a topological perspective, but also in terms of its inherent function, namely dressing. Therefore, a robot interacting with clothes should, in some way, model the garment openings. In particular, a specific opening represented by boundary component $b_0$ has been identified as being essential during robotic assistance with dressing.

*Representation of the garment interior:* For some clothes manipulation tasks, such as folding or flattening, it may be sufficient to only represent the geometry of the garment surface. However, in more complex interactions involving openings, such as sliding a limb or another object through a piece of clothing, the interior of the garment plays an important role. Consequently, it may be useful to have an explicit representation of how the openings are linked inside the garment.

*Integrated kinematic model:* In order to manipulate clothes, the robot also has to coordinate its own mechanical components. The fact that during dressing assistance the robot hands sometimes need to be attached to a human limb further complicates the situation. This is, in some sense, similar to integrating a complex tool into the kinematic chain of the robot, which is not an exclusive problem to robot-assisted dressing. For example, an integrated kinematic model of the robot and a clothes hanger might facilitate sliding the hanger through a garment opening.

*Representation of knowledge gained from experience:* While a predefined plan is well-suited for specifying how to proceed after reaching a subgoal, finding a policy for accomplishing a subtask, such as pulling a garment over a body part or another object, may require trial and error learning. Therefore, the robot has to store, e.g., trajectories that it found to be successful during its exploration. But beyond that, it also has to remember and take into account negative experiences in order to imporove gradually.

## 2.4 Strategies for Reducing and Searching the Problem Space

Given the vast amount of degrees of freedom most garments have, a naive (i.e., exhaustive) internal representation of the clothes perception and manipulation problem would be exceedingly large. To model the state of the environment in its entirety, both its configuration and the complex motions would have to be represented. Furthermore, one would have to integrate a model of the forces acting within the clothing material and the contact forces between the involved objects. Even if the robot had access to all these data (which is far from the case as we have seen), it would be almost impossible to find solutions in such a high-dimensional and unstructured problem space.
But it is not only the size of the state space but also the action space that complicates the search. Take the simple example of sliding a piece of clothing

across a support surface. The robot could use one or more fingers, change the contact position and angle, alter the direction of movement, or vary the force it applies to the garment. In other words, it has so many options that it seems difficult to even determine where to begin. Therefore, we think that robotic action should be tightly coupled with perception. Besides, we suggest that the size and seeming randomness of the problem space of interaction with clothes be reduced by using more qualitative (topology-based) spatial representations. As another strategy for simplifying the problem, we discuss ways to transition from high-dimensional task-agnostic state and action spaces to lower-dimensional task-centric policy spaces that are much easier to search.

### 2.4.1 Integrating Action with Perception

The set of actions available to the robot for solving a given problem has a large effect on the optimal search strategy through the problem space. In Section 2.1.2, we have considered interaction with clothes in isolation from perception. But the actions/operators in the problem space theory may be different from the described interaction primitives in that they are supposed to provide ways to move from one *perceived* state to another until a goal state is reached. Therefore, both the state representation and the overall problem to be solved have to be taken into account when specifying the operators. If, for example, the robot is faced with a high-level problem such as doing the laundry, the action space should, in the first instance, also be made up of high-level operators (sort the clothes by material and color, put them in the washing machine, etc.). By contrast, if the task is to grasp a piece of clothing and the state representation is derived from depth information, the lower abstraction level should be reflected in the specification of actions, too (e.g., move the fingertips of the right hand toward a graspable point on the garment surface extracted from a point cloud of the scene). Generally speaking, when defining the action space, we try to find the minimum set of operators such that, considering the perceptual abilities of the robot and the internal representation of the task, a sequence of these operators can be found that solves the given problem.

In robotics, it is sometimes implicitly assumed that the operators are *physical* actions and the role of perception is only to update the internal state representation. But it is often the case that perception itself is the problem to be solved (or at least an essential subproblem of a wider task). Then, the operators may be search steps within the state space which do not change the task environment in any way. For example, in visual tracking of clothes, the goal at each time step is to find the garment configuration that, given the

configuration at the previous time step, explains the current visual input best or, more precisely, good enough according to some criterion. The operators required to reach the goal state defined in this way may be simulated rotations, translations, and deformations that iteratively transform the internal garment model.

However, not only simulated actions but also real physical interaction with the environment can be helpful for solving perception tasks. This is referred to as *interactive perception* in the literature (e.g., [12, 13]). Many robot control architectures consist of three elements which are repeatedly applied in the following order: (i) sensing, (ii) planning, and (iii) acting. Traditionally, the second and third steps are based on the first one, i.e., the purpose of perception is to enable the robot to plan its actions. Interactive perception reverses the causality, i.e., the robot acts in order to improve its perception of the environment.

In the clothing domain, the interactive perception paradigm is particularly useful because it can help overcome some of the issues of visual perception mentioned in Section 2.1.1. Consider the task of counting the items in a heap of clothes. While the segmentation problem is practically impossible to solve based on images of the static scene, interacting with the environment by randomly picking and removing one item after the other makes the task much easier to accomplish. In a similar way, the self-occlusion problem in recognition and pose estimation can be tackled by turning a garment such that the relevant parts become visible to the camera.

Interestingly, the issues that have arisen in the context of affordances (Section 2.1.3), namely separating true from false affordances and making hidden affordances perceptible, are both problems that can be solved using interactive perception. For example, in order to determine whether a breast pocket is real or just printed on a shirt, the robot could try to pull at the perceived hem and see if it creates an opening. Similarly, to uncover the hidden pocket of a crumpled shirt, the robot could stretch or flatten the garment.

### 2.4.2 From Geometry to Topology and Back

Representing the complex geometry of the involved objects (particularly the garments themselves) in detail blows up the problem space of interaction with clothes extremely. Therefore, this space should be reduced in size by only modeling the most relevant features of the environment. Since we have shown that, at least on a qualitative level, clothes are aptly characterized by topology, it seems natural that feature selection should be based on topological considerations.

If biological evolution had given rise to representation structures in the brain that are consistent with the mathematical notion of topology, this would strongly support our argument. In fact, researchers have speculated that topological representations might dominate geometric representations in humans. Piaget and Inhelder [14] found that children at a particular stage of development succeed in drawing a cross ("+") while a hand-drawn circle is not distinguishable from a triangle or a square. This indeed corresponds with the definition of topologically equivalent (homeomorphic) spaces. However, Darke [15] remarks that topological primacy in representation is not the only possible explanation of this phenomenon because a child's drawing is influenced by sensorimotor abilities as well as likes and dislikes, and the end product of a drawing is often less revealing than the process of creating it. Moreover, parts of Piaget and Inhelder's work have been criticized for being difficult or impossible to replicate and for using the term *topology* loosely [15, 16].
Our approach to topology-based modeling is not so much motivated by psychological findings on how humans generally represent space as by a task-driven perspective on how robots should represent spatial knowledge to solve problems related to handling clothes. Because garments are usually highly deformable and topology is concerned with the properties that still hold if an object is continuously deformed, the most general yet relevant features of clothes are almost inevitably the topological invariants. However, we think that topology can and should not replace geometry but provide a means for reconciling bottom-up and top-down perception.
The input to the robot's perceptual system tends to be of a geometric nature because sensors can only make quantitative measurements. For example, the depth cameras we have used in many of our experiments measure distances of objects from the camera. Usually, this data travels through a bottom-up processing pipeline to gradually yield more qualitative information about the environment. At the same time, the robot has some abstract prior knowledge that it can use to perform a more targeted top-down search for task-relevant structures in the data.
We believe that, in the clothing domain, both pathways often meet at the level of topology. However, to control its movements, the robot has to rely on exact positional information. Therefore, topological representations have to be enriched with geometric data, again. In this sense, our strategy for reducing and searching the problem space can be described as going from geometry to topology and back. We give two examples of how our robot uses this strategy to perceive clothes:

*Example 1:* In robot-assisted dressing, a combination of bottom-up and top-down processing can be used to find the garment opening through which

the human limbs have to pass. On the one hand, the bottom-up vision pipeline contains an edge detection step and a procedure for representing the edges topologically in a graph[3]. On the other hand, it is known in advance that a garment opening is edged in a certain way, and that the corresponding boundary component is a 1-manifold, i.e., a closed curve (Section 2.2.1). The robot can apply this prior knowledge to the graph in a top-down way in order to find the patterns that yield the desired boundary component and thus the garment opening. However, to determine if a limb fits through the opening, topological knowledge is not sufficient, but the size of the opening has to be computed geometrically.

*Example 2:* If the problem is to classify a piece of clothing that lies spread out on a support surface, an important step of bottom-up processing is to extract the contour of the garment from the input image. This information together with the robot's prior knowledge about prototypical garment shapes can be exploited for classification. Beyond that, knowing the garment category usually implies knowing the number of openings. But again, to be useful, this topological knowledge has to be augmented with geometric data, e.g., with the exact positions of the openings.

### 2.4.3 Task-centric Policy Spaces

We recall that the problem space theory covers three major aspects of problem solving by an information processing system: (i) The environment is represented in such a way that solutions can be found by searching in a space of internal states. (ii) The problem solver chooses its actions depending on the current state. (iii) The actions are goal-directed. Although the focus of the problem space theory has initially been on human problem solving, it also provides an interesting perspective on the question of how difficult a given task is for a robot. However, in order to be useful for robotic decision making in a wide range of problems, it lacks the important concepts of uncertainty and motivation which are made explicit by another theory, namely the theory of *Markov decision processes (MDPs).* MDPs add to the set of states and the set of actions a probability that choosing a certain action while being in one

[3]Intuitively, a graph is a topological representation in the sense that it focuses on neighborhoods rather than distances. Formally, we have discussed in Section 2.2.1 how the realization of a graph as a simplicial 1-complex can be considered a topological space, using the example of what we have referred to as a textile graph. We emphasize however that, while the concept of textile graphs has been introduced to formalize the objective spatial structure of garments, using graphs during perceptual processing serves to organize the robot's subjective knowledge, which are two completely different things.

state leads to being in another state in the next time step. Moreover, they introduce the notion of a reward that the robot receives for transitioning from one state to another by taking a particular action.
In an MDP, the solution to a problem is represented by a so-called *policy*, a function specifying which action to choose (or the probability of choosing an action) when being in a certain state. In episodic tasks and if the complete state of the environment cannot be observed, instead of focusing on the values of individual states and actions, it is often more efficient to search in a space of parameterized policies (e.g., in a space of parametric trajectories) directly, using an evolutionary algorithm or a similar optimization method. Then, the reward does not specify the immediate feedback for taking an action in a particular state but the overall performance feedback the robot receives for a full episode of the task, and uncertainty is not necessarily considered on the level of state transitions but only on the reward level.
Regardless of whether one sticks to MDPs or opts for evolutionary algorithms (as we do in our experiments), one has to decide on the structure of the internal task representation. Some approaches to robotic problem solving try to be, at least in principle, task-agnostic by working with raw sensory data such as pixel values from camera images or the joint angles of a robot arm. Most *deep learning* algorithms belong to this class of methods. They learn intermediate representations from a vast amount of training data in order to avoid manual feature engineering. At the other end of the spectrum are those methods that use a lot of domain knowledge to model the relationship between the sensory input and the internal representation explicitly.
We believe that the optimal abstraction level of the representation is highly dependent on the task at hand. The deep learning paradigm has been used successfully for image recognition [17], robotic grasping [18], as well as for playing Atari [19] and the game of Go [20], to name but a few applications. However, in the domain of garment perception and manipulation, there are good reasons for employing task-specific representations rather than relying on raw input data. In the following, we express some concerns with respect to using present-day deep learning methods as black boxes for interaction with clothes and suggest strategies for using task-centric policy spaces instead.

*Sample efficiency:* Deep learning tends to be extremely data hungry. For example, the *ImageNet ILSVRC* dataset [21] that has been used successfully for training deep convolutional neural network classifiers [17] contains 1.3 million labeled images and 1000 object classes. Even if the network architecture and the hyperparameters are reused across similar tasks, e.g., across different Atari games [19], the actual learning procedure has to be carried out separately for each task. In robotic interaction

with clothes, it would be very difficult and costly (if not impossible) to generate thousands of training samples. Especially if human subjects are involved as in robot-assisted dressing, such an amount of training sessions would be inacceptable. Therefore, in scenarios of that kind, the dimensionality of the policy space should be minimized by using task-specific heuristics so as to reduce the number of samples required.

*Integrating prior knowledge:* To be more or less independent of the task (if only in theory), most deep learning methods use as little domain knowledge as possible. Consequently, there is typically no easy way to integrate prior knowledge into these frameworks. However, as we have seen in our analysis of the problem, robots have to exploit very abstract topological concepts (e.g., that of garment openings and how to make use of them) to handle clothes successfully. But it is difficult to imagine that a deep learning system could develop such a qualitative understanding of the task from scratch because the solutions learned by most existing algorithms are more superficial than they might initially appear, as pointed out by Marcus in [22]. By contrast, in task-centric policy spaces, robots do not acquire new concepts, but they learn how to apply known concepts optimally, which is usually much easier.

*Hierarchical planning:* Many tasks in the clothing domain, e.g., robotic assistance with dressing, have a hierarchical structure. However, the correlations between non-hierarchical feature sets learned by present-day deep learning algorithms typically cannot represent such structures in a natural way [22]. Moreover, these methods often fail to find strategies that extend over longer time scales [19]. Therefore, to carry out complex tasks, our robot follows a high-level plan and uses policy learning only for solving particular subproblems. When designing the policy space, we can thus assume that the environment is in a certain initial configuration, and that the robot has a clearly defined goal, e.g., to optimize the end effector trajectory for the specific subtask of pulling a piece of clothing over one of the human limbs during dressing assistance.

*Dealing with rare events:* To acquire comparatively simple manipulation skills such as grasping objects [18], robotic learners can rely on receiving reward at regular intervals. This is even more the case in computer games with an ever-increasing score. In poorly initialized deep learning of interaction with clothes, it would take a very long time before the robot does anything meaningful and reward-worthy at all. During this exploration phase, the robot would behave almost randomly and it would be difficult

to guarantee the safety of the involved objects and/or persons. By contrast, task-centric policy spaces can be limited to at least theoretically plausible policies for accomplishing a certain task. Beyond that, it may be possible to exploit the task-specific structure of such policy spaces for defining objective functions that do not only reward the rare event of complete success but guide the robot to successful policies gradually (task-centric reward shaping).

## 2.5 Discussion

In this chapter, we have developed a problem definition of bimanual interaction with clothes, with a view to finding suitable ways of representing the problem in robots. This has been done within the framework of the problem space theory. The theory states that any problem solving strategy can be reformulated as a search through a space that constitutes the robot's internal representation of the task. We point out however that, throughout this thesis, we will not always make the problem space (i.e., the state space, the action space, as well as the initial state and the goal state) explicit. Moreover, at times, it will be clear that the robot performs a search (e.g., a stochastic search for an optimal policy), whereas at other times, it will be more implicit. Notwithstanding this, the problem space theory has been helpful in understanding the challenges of clothes perception and manipulation, and in identifying strategies for simplifying the task by reducing the search space complexity and size.
We began our analysis by considering robotic perception in the clothing domain. In other words, we regarded the link between the task environment and its internal representation. We found that, in this context, a robot faces two key challenges, namely to reconstruct the configuration of garments and to detect their physical interaction with other objects indirectly from the perceptible effects. It has become apparent that both problems can, in principle, be solved using visual perception. However, the situation is complicated by the fact that clothes tend to be extremely deformable which can result in prohibitively high-dimensional models. On the other hand, the deformability of garments also affords a large amount of interaction possibilities with them. These interactions can be useful in two respects. For one thing, they possibly change the state of the environment such that it progresses toward a state that corresponds to a goal state in the problem space. For another thing, interaction may simplify and improve the robot's perception of the scene, which is often referred to as interactive perception.
There is a related controversy about whether perception should be a bottom-up process starting at the environment affordances or a top-down process guided

by prior knowledge about the task. We think that information should travel in both directions, eventually converging to qualitative representations that, in the case of interaction with clothes, are often aptly described at the level of topology. From the spatial structure of garments, we have derived that the most important topological invariant of clothes is the number of openings or, more formally, the number of boundary components.
The openings are not only essential in characterizing garments topologically but also in view of the inherent purpose of clothing which is to cover the human body. We have described the task of putting on clothes both in an abstract topological manner and more concretely with regard to robot-assisted dressing. Based on this functional analysis, we have made some suggestions for how garments and the other entities involved in the task can be represented in robots. Furthermore, we have argued that the structure of the task should also be considered and exploited when defining policy spaces for robotic motor skill learning in the clothing domain. This is opposed to the vast majority of deep learning methods that deliberately ignore any domain knowledge to be, at least to some extent and in theory, task-agnostic.

# 3 Related Work

One guiding principle of this thesis is to take an integrated view of perception and action. Another central idea is to use concepts from topology for more qualitative representations of the environment. Both ideas are shared by a growing number of researchers in the field of cognitive robotics. Therefore, in this chapter, we will first give an overview of works that are concerned with the question of how action can facilitate perception and vice versa (Section 3.1). Then, we summarize the related work on topology-based representations in robots (Section 3.2). In Section 3.3, we provide a literature review of robotic clothes manipulation. Finally, since the dressing task has been identified as being essential in characterizing the form-function relationship of garments, we take a closer look at recent works on robotic dressing assistance (Section 3.4).

## 3.1 Coupling Perception and Action

Perception and action can be coupled in two ways. It is widely accepted that perception is often a requirement for meaningful action selection, e.g., in the *sense-plan-act* paradigm, in *visual servoing*, and whenever a state representation of the environment is needed (such as in *Markov decision processes*). However, in the following, we focus on the opposite case in which action facilitates perception. For a detailed survey of *interactive perception*, the reader is referred to [13]. The authors distinguish interactive perception from *active perception* [23] which does not involve forceful interaction with the environment. Examples of active perception include *simultaneous localization and mapping (SLAM)* [24] in which a robot has to move around its environment in order to build a map and at the same time localize itself in it, and *active vision* [25] in which the robot actively changes the viewpoint(s) of its camera(s) but leaves the rest of the environment unchanged.
If touch is the main modality considered, it is quite clear that interaction with the environment is almost always necessary to localize [26], explore [27], or identify [28] objects. Moreover, Li et al. [29] showed that, using haptic exploration primitives and a tactile servoing controller, it is possible to learn the homogeneous transformation of a tool with respect to the robot's end effector.

Meier et al. [30] demonstrated that push manipulation is useful in classifying the behavior of an object (sliding vs. slipping) during interactive perception with a tactile fingertip. Vásquez et al. [31] suggested robotic grasping of an object to identify its shape by means of a so-called *invariant proprioceptive signature* based on the joint angles of the hand.

In recent years, there has been an increasing number of works investigating how interaction can be used to improve *visual* perception. In an influential paper of Katz and Brock [12], who coined the term interactive perception, the authors describe a system observing changes in distance between points on the surface of an articulated object during robotic manipulation so as to build a kinematic model of the object. The problem of estimating articulation models using interactive perception has since been addressed in a number of projects (e.g., [32–34]). Other visual perception tasks that have been approached with an interactive method include segmentation (e.g., [35–37]) and object recognition (e.g., [38–40]).

Interactive perception and regrasping can be combined such that, on the one hand, moving a grasped object in front of the robot's camera improves perception, and on the other hand, the improved model of the object helps to specify suitable poses for regrasping. Krainin et al. [41] suggested an information gain based next best view algorithm guiding manipulator motion and object regrasping from the tabletop so as to minimize uncertainty in object shape. Tsuda et al. [42] extended the idea to the bimanual case, i.e., they considered in-air regrasping. In both works, the goal was to generate complete volumetric or mesh models of unknown rigid objects.

Interactive perception has also played an important role in robotic perception and manipulation of deformable objects such as clothes. Sun et al. [43] and Willimon et al. [44–46] explicitly characterized their works on robotic laundry handling as applications of the interactive perception paradigm. However, many projects have used similar ideas. For a detailed review of works concerned with perceiving and handling garments, the reader is referred to Section 3.3.

Another concept that relates action to perception is that of affordances. In Şahin et al.'s formalization [47], an affordance from the robot's perspective is a relation between an entity-behavior tuple and its effect, written as $(< \textit{effect} >, < (\textit{entity,behavior}) >)$, where angle brackets denote equivalence classes. In this framework, a key aspect of learning is forming these equivalence classes, i.e., abstracting from irrelevant properties of the environment. For example, the effects of lifting a red object with the right hand and lifting a blue object with the left hand are equivalent. *Object-Action Complexes (OACs)* [48] are another formalization of affordances as state-transition functions. The focus of OACs is not on equivalence classes, but on the fact that actions can

also be unsuccessful. Specifically, an OAC is a triplet $(E, T, M)$, where $E$ is an identifier, the prediction fuction $T : S \rightarrow S$ represents the system's belief about how the OAC will change the state of the environment if successful, and $M$ is a statistical measure of the OAC's success in the past. In general, this framework can be used for planning at different levels of a cognitive architecture (from high-level reasoning to the sensorimotor level) and for learning $S$ (the state representation), $T$, $M$, or a control program that maximizes the probability of success.

## 3.2 Topology-based Representations

An aspect we and other researchers have added to the affordance discussion is the relationship between the action possibilities of an object and its topological invariants. Under the assumption that topological representations facilitate the coupling of perception and action, roboticists have exploited various ideas from topology. Since some of the concepts used are related but not identical, we begin this section with a few definitions in order to avoid confusion.

*Homology group:* A homology associates to an object, such as a topological space, a sequence of homology groups. Roughly speaking, the elements of the homology groups $H_k(X)$ with $k \in \{0, 1, ..., n\}$ of an $n$-manifold $X$ are equivalence classes of $k$-dimensional cycles (looping around $k$-dimensional holes) [1]. If $X$ is a connected and orientable 2-manifold without boundary of genus $g$, the rank of $H_0(X)$ is 1, the rank of $H_1(X)$ is $2g$, and the rank of $H_2(X)$ is 1.

*Winding number:* The winding number is a topological invariant of a closed curve in the plane with respect to a particular point (usually the origin). It is an integer that counts the number of times the curve travels around the point, and it does not change if the curve is continuously deformed without traversing the point. Given a closed, differentiable curve $\gamma$ with points $(x, y)$ along the curve, the winding number around the origin can be defined as follows:

$$W = \frac{1}{2\pi} \int_{\gamma} \frac{x\ dy - y\ dx}{x^2 + y^2} \tag{3.1}$$

[1] In practice, the construction of the homology groups of a manifold often begins with defining a simplicial complex on the manifold (triangulation) so that *simplicial homology theory* can be applied. Then, a *k-chain* is a linear combination of $k$-simplices and a *cycle* is a chain without boundary. Two $k$-cycles $z_1$ and $z_2$ are equivalent if $z_1 - z_2$ is the boundary of a $(k+1)$-chain (negation meaning inverting the orientation), and a $k$-cycle is trivial (neutral element of the $k^{\text{th}}$ homology group) if it is a boundary all by itself.

It is also possible to compute a winding number for a piecewise linear and even a non-closed curve. Then, however, $W$ is no integer and no topological invariant anymore, but it is still a measure of the curve's winding around the origin.

*Linking number:* Given two non-intersecting simple, closed, differentiable curves $\gamma_1$ and $\gamma_2$ with points $r_1$ and $r_2$ along the curves, the linking number can be computed by the following Gauss linking integral:

$$Lk = \frac{1}{4\pi} \int_{\gamma_1} \int_{\gamma_2} dr_1 \times dr_2 \cdot \frac{r_1 - r_2}{|r_1 - r_2|^3} \tag{3.2}$$

The definition is very similar to that of the *writhe* (Section 5.2.3). However, while the writhe is defined for a single curve and takes values in $\mathbb{R}$, the linking number is an integer and a topological invariant of two closed curves. Like the winding number, it can also be computed for piecewise linear and non-closed curves, taking values in $\mathbb{R}$ and losing its property of being a topological invariant.

*Writhe matrix:* Ho and Komura [49] introduced the concept of a writhe matrix for two (possibly non-closed) piecewise linear curves. Unlike the name suggests, it is most closely related to the linking number. Specifically, each entry of the matrix represents the (real-valued) linking between two line segments of the curves.

*Topology coordinates:* In the same paper, the writhe matrix was used to define the so-called topology coordinate system describing how two curves are tangled with each other. The first coordinate is the sum of the matrix entries (which is equivalent to the real-valued linking number of the curves) and indicates the amount of tangling. The second and third coordinates represent the center of mass of the entries and correspond to the center of the tangled area. The fourth coordinate can be determined by computing the orientation of the matrix entries' principal axis and corresponds to the density of the tangled area.

*Reeb graph:* The basic idea of *Morse theory* is to study the topology of a manifold with respect to a differentiable function defined on it. One of the most popular object representations originating in Morse theory is the Reeb graph which describes the evolution of the function's level sets on the manifold. Specifically, the vertices of a Reeb graph correspond to the critical points of the function (where the level set topology changes), and the edges represent connected components of the level sets.

Pokorny et al. [50] suggested a topology-inspired approach to robotic grasping of objects with holes. They share with us the idea that finding certain closed curves on the surface of an object is an important factor for meaningful and successful grasping. However, they focused on 2-manifolds without boundary, so that the most relevant closed curves were not boundary components but the one-dimensional cycles from the first homology group (which loop around one-dimensional holes). In particular, they built simplicial complex representations of the objects considered before employing the method from [51] to find a basis of $H_1(X)$ that consists of loops representing the shortest cycles in their respective equivalence classes. Then, they used the real-valued winding number for non-closed curves to optimize how the control curves (e.g., the curve connecting the finger tips of the thumb and the forefinger) wind around a target point on an object loop during grasping. Finally, they exploited another topological concept, namely the real-valued linking number between the control curves and the loop, for evaluating the caging grasp after execution. In [52], very similar ideas were used and extended to the interaction primitives of clasping, latching, and hooking. Winding numbers were also shown to be useful for grasp transfer between different hand kinematics [53].

*Persistent homology* describes how a topological feature, in particular homology, changes if the spatial resolution of an object (considered as a topological space) is changed. In [54], persistent homology was used to classify robot trajectories into equivalence classes such that no trajectory of one class can be continuously transformed to a trajectory of another class. Beksi and Papanikolopoulos [55] suggested a signature of topologically persistent points as a global point cloud descriptor.

Writhe matrices and topology coordinates were initially used in the context of character animation [49]. Recently, the idea was taken up by several roboticists. Ho et al. [56] demonstrated how a humanoid robot can be controlled in topology coordinates. Zarubin et al. [57] made use of the concept in hierarchical motion planning. Vinayavekhin et al. [58] considered the problem of how a robot can learn to regrasp objects from human demonstration. They used topology coordinates to express the tangle relationship between the hand and the object to be grasped. Yuan et al. [59] presented an approach to reinforcement learning of robotic whole arm manipulation skills for moving a human body, e.g., in a swimming rescue scenario. To represent the topological relationships between the robot arms, the human arms, and the torso, combined writhe matrices were used. Topology coordinates were also applied to robot-assisted dressing [60, 61], as described in Section 3.4.

The key ingredient for constructing Reeb graphs is the real-valued function defined on the manifold considered. For illustration purposes, the height func-

tion is often used. Then, however, the Reeb graph is not rotation invariant, i.e., it might look different depending on how the manifold is oriented in space. Therefore, Berretti et al. [62] used the $AGD$ function (i.e., the average geodesic distance of a point to all the other points on the object surface) in their algorithm for decomposing objects into parts using Reeb graphs. The method was applied to part-based robot grasp planning from human demonstration [63] and to the robotic task of handing over an object to a human in a comfortable way [64].
Regardless of how they are created (using Reeb graphs or any other topological or non-topological technique), skeletons can be considered topology-based representations in the sense that they try to reduce complex geometric shapes to connections between the most relevant parts. Among other things, skeleton models were used for shape matching [65] and as simplified representation of the human body during tracking and recognition [66]. In a wider sense, any model that is focused on adjacency rather than distance, such as a graph or a topological map [67], can be considered a topological representation.

## 3.3 Handling Clothes

It is only in recent years that roboticists have begun to focus on objects with very high-dimensional configuration spaces. While ten years ago, robotic handling of clothes was still quite underinvestigated, the amount of publications in this field of research has since increased tremendously. Perception and manipulation tasks studied include clothing classification [43–45, 68–75], pose estimation [76–81], grasping [82–88], unfolding [46, 89–94], flattening [95, 96], ironing [97, 98], and folding [99–105].
Willimon et al. [44] suggested using interactive perception for the recognition of garments. Their robot first extracted an item from a pile of laundry before classifying it based on images obtained from a procedure of repeated dropping, regrasping, and observing. In a later work [45], a multi-layer classification strategy was introduced to improve the accuracy of the method. Sun et al. [43] proposed another manipulation-perception cycle to gradually increase the confidence of a probabilistic classifier. In [68], they also presented an algorithm for single-shot recognition of the clothing category. In the experiments of Hu and Kita [69], the robot brought the garment into one of a limited number of shapes to facilitate classification on the basis of the object contour. The Gabor filter based image features suggested by Yamazaki and Inaba [70] provide classification-relevant information about the clothing material, wrinkles, and folds. Kampouris et al. [71] proposed an explorative approach to garment perception using RGB-D, tactile, and photometric stereo sensors. Besides rec-

ognizing the garment category, their method is capable of classifying the fabric pattern and material.

While it is still an open question whether black box deep learning is suitable for end-to-end learning of complex garment handling tasks, a few researchers have been able to show that particular deep learning methods can be used to solve essential subproblems such as clothing classification. Gabas et al. [72] proposed a *convolutional neural network (CNN)* approach to occlusion-robust garment recognition using depth information. In [73], the system was expanded by two additional CNNs performing a search for relevant grasping points. Mariolis et al. [74] presented another CNN-based framework containing two layers, one for category recognition and one for pose estimation. Yuan et al. [75] suggested an active exploration approach to garment perception and classification based on material properties such as thickness, smoothness, and textile type, using tactile sensing and deep learning.

Cusumano-Towner et al. [76] proposed a *hidden Markov model (HMM)* for estimating the configuration of an article of clothing during a sequence of specific manipulations and observations using a rather simple observation model. Kita et al. [77] suggested the following two-stage method for pose estimation: First, a number of representative shapes is generated through physical simulation of hanging clothes. In the second stage, the shapes are deformed to match the observed data and select the most consistent hypothesis. Beyond that, the authors showed in [78] that cues from strategic observation can be helpful for garment state recognition. In a similar way, Li et al. [79] first generated training data by simulating hanging garments grasped at different points and capturing depth images from different views. Then, using *sparse coding* and *SVM* classification, they were able to predict both the object category and the grasping point (and thus the object pose). In [80], the method was improved both in accuracy and speed by using volumetric fusion instead of individual depth images and majority voting. Wang et al. [81] restricted themselves to algorithms for the perception of socks. Their robot was able to predict the configurations and to match similar socks into pairs.

The focus of Ramisa et al.'s work has been on robotic grasping of clothes. In [82], they proposed a measure of wrinkledness computed from the distribution of normal directions in a point cloud. In a later work [83], appearance features were used in addition to depth information to select optimal grasping points. Moreover, they built a 3D shape descriptor to characterize textiles and used it for wrinkle detection, instance recognition, and informed robotic grasping [84]. Yamazaki [85] presented an approach to extracting graspable hem elements from depth images. Maitin-Shepard et al. [86] suggested that grasping and regrasping points be selected based on geometric cues like borders

and corners. Gibbons et al. [87] addressed the problem of identifying possible grasping locations on a pile of clothes in a remote manipulation scenario. Their method takes into account visual features as well as the kinematic abilities of the robot. Since different techniques and experiments in the field of robotic grasping of clothes are difficult to compare, the reader is referred to [88] for a method of characterizing grasping systems at the level of perception-action couples.

Unfolding a crumpled piece of clothing is often considered as a preliminary to folding. In an early work, Hamajima and Kakikura [89] suggested a heuristic strategy with the aim of grasping the garment at two hemline points, followed by a classification and a shaping step. In a similar spirit, Triantafyllou et al. [90] presented a geometric approach to the task of unfolding clothes based on the idea that a hanging garment held by two points can be approximated as a planar object. Yuba et al. [91] showed that simple pinch and slide motions can be used for unfolding. However, their method is limited to rectangular pieces of cloth. Doumanoglou et al. [92] approached the unfolding problem with some reasoning about possible lowest points together with *random decision forest* based recognition and probabilistic planning. Li et al. [93] applied their recognition scheme [80] to the unfolding task, optimizing an evaluation function of a two-point grasping configuration through iterative regrasping. The robot in [46] unfolded a piece of laundry by pulling it in different directions at various points using interactive perception. Stria et al. [94] proposed a method for bimanual robotic unfolding of an item of clothing placed flat on a table and folded over a particular axis.

If the item of clothing considered is only moderately wrinkled, it may be sufficient to flatten the garment rather than to completely unfold it. The method described by Sun et al. performs a surface analysis of a 2.5D heightmap representation of a garment and was used successfully for robotic dual-arm [95] and single-arm flattening [96]. Flattening an article of clothing is also the goal of the ironing task, which was addressed by Estevez et al. [97] combining 3D perception with force/torque sensing. Besides, Li et al. [98] suggested a multi-sensor surface analysis fusing a curvature scan and a discontinuity scan and applied it to robotic ironing.

There have been a few studies concerned with the problem of folding a garment that was more or less crudely spread out on a flat surface. Van den Berg et al. [99] presented an algorithm planning a sequence of gravity-based folds referred to as *g-folds*. Bersch et al. [100] proposed a fold detection, grasp generation, unfolding, and folding strategy that was applied to clothes with printed fiducial markers simplifying perception. Both Miller et al. [101] and Stria et al. [102] initialized 2D polygonal garment models and used these models to implement

a robotic folding procedure. In a work of Kita et al. [103], a folding task was implemented in a humanoid robot using the recognition algorithm described in [77]. In two different papers, the problem of trajectory optimization preventing garment slipping during folding is addressed. While Li et al. [104] employed a physics engine, Petrík et al.'s model is based on the equilibrium of forces and assumes a rectangular piece of cloth for simplicity [105].

## 3.4 Robot-assisted Dressing

Over the last few years, robot-assisted dressing has attracted increasing interest. While solutions for the general task are still out of reach, there is a growing number of works that demonstrate how important parts of the problem can be solved. In many cases, this is done by exploiting simplifications gained through focusing on a narrow subdomain.
Most projects in the field of robotic dressing assistance make use of machine learning or stochastic optimization techniques. Then, a key distinguishing characteristic is the objective function used because it determines what is to be optimized. Colomé et al. [106] proposed a reinforcement learning framework for the problem of wrapping a scarf around the neck of a mannequin. Since no tight garment openings are involved in this particular task, it was possible to minimize a rather simple objective function using the spatial distance of the scarf from a reference position. In [107], a reward-weighted *Gaussian mixture model (GMM)* was proposed for action selection in a robotized shoe dressing task, the reward reflecting success/failure. Gao et al. [108] suggested a path optimization algorithm for putting on a jacket without sleeves. In this task domain, an objective function that aims at avoiding large external forces and, in this way, external resistance has proven to be effective. Tamei et al. [60] modeled the relationship between a T-shirt's neck opening (equipped with markers) and a mannequin's head through topology coordinates which were used to define a reward function. In [61], the authors have reported some success in markerless estimation of the coordinates.
While many studies on robot-assisted dressing focus on trajectory planning, there is also a number of works concerned with user modeling during the dressing process. In [109], random decision forests were employed to estimate the upper-body pose and GMMs were used to model the movement spaces of particular body parts. Chance et al. [110] used *recurrent neural networks (RNNs)* to predict the elbow position based on other features of the user pose under occlusion. The two key contributions of Zhang et al.'s work [111] are (i) a control strategy that minimizes forces applied between the user and the robot and (ii) the use of a *Gaussian Process Latent Variable Model (GPLVM)* for

modeling the user movement limitations and updating the dressing trajectory accordingly. Clegg et al. [112] used reinforcement learning to model what a human character model is capable of doing in a simulated robotic dressing assistance scenario. In [113], a method for tracking the user pose during robot-assisted dressing with capacitive proximity sensing was described.
Haptic information can be used to learn how to navigate a simulated arm through the interior of a garment, as shown in [114]. However, in more realistic dressing assistance scenarios, contact between a garment and the human body can often be inferred only indirectly from feedback at the robot's end effector. Kapusta et al. [115] distinguish three possible outcomes of the task of pulling a sleeve over a person's forearm: The hand misses the opening to the sleeve, the hand or forearm gets caught in the fabric, or the full forearm successfully enters the sleeve. Their algorithm is able to classify these three outcomes using forces measured at the end effector. Yu et al. [116] employed a simulator to reduce both the classification error and the amount of required real-world samples. Erickson et al. [117] were even able to infer the areas of contact between the sleeve and the arm in simulation, and to use the approach in a real robot-assisted dressing task [118]. The work of Clegg et al. [119] is a good example of how high level planning and deep learning techniques can be combined by separating the dressing task into subtasks and learning a control policy for each subtask, even though the method has only been applied to character animation so far.
Furthermore, we should mention that not all works on robot-assisted dressing make use of stochastic optimization or learning. Chance et al. [120] argue that simple *human robot interaction (HRI)* strategies could be used instead of complex machine learning methods. Yamazaki et al. [121] proposed a failure detection and replanning approach using visual and force information in a bottom dressing task. The robot in [122] inferred a knowledge base of user constraints and used it for placing a stiff hat on the head.

## 3.5 Discussion

In this chapter, we have provided a literature review of works that are related to the present thesis in one of two respects. For one thing, we have been interested in how other roboticists tackled the issues of integrating action with perception and of modeling the environment qualitatively in a topological manner. For another thing, we have given an overview of methods whose area of application is robotic interaction with clothes.
We have seen that there is a large body of research on interactive perception. The approaches discussed range from haptic exploration to interactive robot

vision and regrasping-perception coupling. Moreover, the idea of object affordances as action possibilities provided to the robot has proven beneficial in congnitive robotics. Therefore, attempts were made to formalize the concept. However, these formalizations aim at specifying the meaning of affordances in general rather than aiding in finding the affordances of particular objects. We believe that the topological properties and the affordances of an object are strongly correlated, and it has become apparent that other researchers in the field have had similar ideas. Action-relevant invariants used by robots include homology, winding and linking numbers, topology coordinates, and Reeb graphs. But to the best of our knowledge, boundary components and their topological connections in the interior of an object have not been considered so far. Furthermore, the idea of using interactive perception to build topology-based object representations appears to be largely unexplored.
Grasping, folding, unfolding, and flattening are some of the most extensively investigated tasks in the domain of robotic garment manipulation. These tasks have been approached with a wide variety of methods including interactive perception, RGB-D vision, tactile sensing, physical simulation, deep learning, and traditional machine learning algorithms. When comparing with the taxonomy of interaction primitives we have developed in Section 2.1.2, it is noticeable that most works focused on primitives from the top right part of Table 2.1, i.e., no garment openings were involved and contact relations to other objects played a minor part. As opposed to this, in the present thesis, we consider primitives from the bottom left part of the taxonomy table in detail. In robot-assisted dressing, the openings almost inevitably play an important role, but most existing works have avoided modeling them explicitly. By contrast, in the following chapters, we will present methods that are relevant to dressing assistance and allow our robot to detect, grasp, track, and manipulate garment openings.

# 4 Reducing the Problem Space for Detection and Grasping

One of the major challenges of handling clothes is the complex dynamics of garments (and textiles in general) because even minimal interaction with such an object can change the parameter values of a physical or geometric model in a seemingly chaotic manner. Consider the introductory example from Section 2.1.1 again: Pushing against a piece of cloth hanging freely under gravity with one finger leads to large changes in configuration, e.g., the cloth may wrinkle up or wrap around the finger in unpredictable ways. However, the object will also return to a state of static equilibrium quickly as soon as the finger stops moving. Therefore, our first simplification of the problem of interacting with garments is to limit robotic perception to the phases in which the environment can be assumed to be static. Of course, this simplification is not always reasonable. For example, in robot-assisted dressing, the dynamics of the garment presumably plays an important part. But under certain conditions, the visual features that are relevant for such essential skills as grasping can be detected in static images of the scene.

We start this chapter with some preliminaries of visual perception by a robot, focusing on complex scenes with multiple objects as they are common in interaction tasks in the clothing domain (Section 4.1). In particular, we describe the sensory input signals mainly used throughout this thesis, namely depth images and point clouds. Furthermore, we discuss how task-relevant objects can be efficiently isolated in such signals (point cloud filtering). Then, we present two different approaches to detecting the openings of a garment which are represented by the boundary components and have been identified in Chapter 2 as being (topologically and functionally) the most important features. The first method makes use of geometric and topological prior knowledge about clothing and assumes that the garment lies spread out on a tabletop (Section 4.2). The second method is based on the interactive perception paradigm and a graph representation of edges. Finally, we show how a bimanual robot can use the information gained about garment openings to find and exploit suitable grasp poses (Section 4.3).

## 4.1 Robot Vision in Complex Scenes

Many algorithms described in this thesis presuppose that it is straightforward to separate the relevant from the irrelevant objects in the robot's field of view. However, even in lab settings, robots often have to cope with more or less cluttered environments. Typical scenes in the domain of interaction with clothes consist of a support surface auch as a tabletop, at least one garment, the robot arms and hands, and some background objects. Additionally, in robot-assisted dressing, there is the human body and perhaps further equipment. In environments of such complexity, visual perception solely based on color images would be very difficult. In static environments, especially in the single-view case (i.e., if the robot has only one static camera), depth perception would be a particularly hard problem because motion parallax and other depth cues that rest upon stimulus changes over time cease to exist. While humans have to rely on this kind of information, many present-day robots are equipped with depth cameras, which have become more and more affordable in recent years. In the following, we briefly outline how these cameras work and describe the properties of the signal they provide directly (depth images) and after preprocessing (point clouds). Furthermore, we present an algorithm for filtering such point clouds semantically and in real time.

### 4.1.1 Depth Images and Point Clouds

Depth cameras are used in various areas including gaming, human-machine interfaces, security, and medical applications. Since Microsoft introduced the *Kinect* sensor (the first high-resolution depth sensor available for less than 150 dollars), and especially since two community projects (*OpenKinect*[1] and *OpenNI*[2]) started to develop open source drivers for reading the sensor data, such devices have also become popular among smaller research institutes and even amateur hackers. In particular, they have been used in many robotics projects. For a review of some early applications of the Kinect sensor, the reader is referred to [123]. There are now a lot of similar (and affordable) camera systems that output depth images, sometimes also called range images, i.e., 2D arrays of pixels, each corresponding to the distance of the nearest light-reflecting object in a certain direction from the camera. The systems use different imaging methods, the most common ones being the following:

*Stereo triangulation:* Using two color cameras, it is possible to simulate binocular depth perception (stereopsis). Just like the human eyes, the cameras

[1] `https://openkinect.org`
[2] `https://structure.io/openni`

need to be located at slightly different (horizontal) positions. Then, from the disparity between two corresponding points in the left and right camera images, a depth value can be computed by triangulation. However, as corresponding points have to be found on the basis of color information, this method may fail for homogeneous parts of the scene, e.g., in the case of single-colored garments or textureless background.

*Time of flight:* In contrast to stereo camera systems, time-of-flight cameras are active sensors, i.e., they do not rely on ambient light but consist of an infrared light source and a receiver unit. The general approach is based on measuring the time the emitted light requires to travel to an object in the scene and back to the camera. Most modern devices, e.g., the second Kinect version (*Kinect v2*), employ continuous wave modulation [124]. In this method, the time of flight is obtained from the estimated phase difference between the emitted and the received signal.

*Structured light:* Depth cameras based on structured light, such as the first Kinect version (*Kinect v1*), combine triangulation with active imaging [124]. Much like in time-of-flight cameras, there is an infrared light emitter and a receiver. However, like in the stereo camera setup, both units have to be positioned at a certain distance from each other to make triangulation possible. Specifically, a known light pattern, such as a structure of lines or dots, is projected onto the scene whose surface shape makes the pattern appear distorted when seen from the receiver's perspective. The depth image is then computed by analyzing the disparity between the original and the distorted light pattern.

In our experiments, we used the *Kinect v1* sensor which provides a depth range of about 500 mm to 4000 mm. The depth camera has a resolution of 320×240 pixels (which can be upscaled to 640×480 pixels), a field of view of 57×43 degrees, and a frame rate of 30 Hz. The pixels take integer values in [0, 2047], the maximum of 2047 representing an invalid measurement. The raw values can be easily converted to real distances with an accuracy of up to 1 mm (depending on the camera calibration) and a precision error proportional to the squared distance[3].
As the sensor uses the structured-light method, there is a horizontal displacement between the infrared projector and the infrared camera which may lead to undefined regions in the depth image. In areas that are in the camera's field of view but on which the light pattern cannot be projected because of occlusion by another (or the same) object, depth estimation is impossible.

[3] `http://wiki.ros.org/openni_kinect/kinect_accuracy`

Our algorithms either ignore these pixels or interpret them as having infinite depth. We note however that specific filters for removing these artifacts have been suggested in the literature (e.g., [125]).
The Kinect sensor provides RGB color images with a resolution of 640×480 pixels along with the depth images. However, there is also a horizontal displacement between the infrared camera and the RGB camera. Consequently, the depth and color images need to be registered onto one another. Moreover, the coordinate systems of the cameras have to be aligned with the world coordinate system (the robot's frame of reference). Both is accomplished by a calibration process estimating the extrinsic (camera position and orientation) and intrinsic parameters (focal length, pixel size, and principle point) of the RGB as well as the infrared camera. We employ the calibration tool provided by the *Image Component Library (ICL)*[4] which uses a 3D calibration object (with a defined geometry and a set of fiducial markers attached to it) that is placed at a fixed position and orientation in front of the camera.
Using the intrinsic and extrinsic camera parameters, it is straightforward to generate a point cloud by converting the pixel values of a depth image to a set of points in world coordinates. In addition to the 3D coordinates, the points may have a color obtained from a depth-registered RGB image and possibly further features such as a label or an estimated surface normal. Point clouds generated from depth images have another advantageous characteristic: Rather than considering them as unordered sets of points, one can explicitly make use of their structure by storing references to the points in a 2D array with indices corresponding to the pixel positions in the original depth image. Such point clouds are referred to as *organized* point clouds and can be used for implementing some algorithms (e.g., algorithms for surface normal estimation [126]) more efficiently by exploiting the additional pixel adjacency information.
Throughout this thesis, we assume that our algorithms have access to the original RGB and depth images as well as a calibrated and color-registered organized point cloud. We use *ICL* not only for camera calibration but also for image and point cloud processing because it allows for easy prototyping, has a rich set of functions, and can be seamlessly integrated with the middleware we use (*RSB*[5] and *ROS*[6]). Moreover, it provides OpenKinect and OpenNI data grabbers as well as wrappers around other computer vision (*OpenCV*[7]), point cloud processing (*PCL*[8]), and parallel computing libraries (*OpenCL*[9]).

[4] `http://www.iclcv.org`
[5] `https://code.cor-lab.de/projects/rsb`
[6] `http://www.ros.org`
[7] `https://opencv.org`
[8] `http://pointclouds.org`
[9] `https://www.khronos.org/opencl`

### 4.1.2 Semantic Point Cloud Filtering

In point cloud processing by a robot, the goal is often to extract certain task-specific patterns not from the entire point cloud but only from the parts representing the relevant objects in the scene. For example, when trying to identify the boundary components of clothes, the sub point clouds corresponding to some background objects may be distracting and only increase the space that has to be searched by the core algorithm. Therefore, to reduce the search space, it is generally useful to remove (or at least mark) the irrelevant points. We make two basic demands on such a point cloud filter. First, using prior knowledge about the shape and pose of the objects in the scene, it should be easy to unambiguously define which parts of the point cloud are to be filtered out. Second, the filter algorithm should be implemented efficiently enough to integrate smoothly with the other vision components. Ideally, the point cloud filter should operate in real time, i.e., it should not hinder the overall processing pipeline from running at the frame rate of the depth camera.

**Functionally Complete Filter Specification** In most scenarios, one can easily express in natural language which are the irrelevant parts of the point cloud. For example, if the robot's task is to recognize an article of clothing placed on a table in a defined workspace, the filter specification could be as follows.

*Example 1: Filter out all points that belong to the tabletop or to the robot or are outside the workspace.*

After detecting the shape and pose of the garment, the robot's task might be to assist a person with putting it on. Then, the following would be a reasonable filter specification.

*Example 2: Filter out all points that belong to neither the garment nor the person to be dressed.*

Using the notation of propositional logic and the set of operators $\{\vee, \wedge, \neg\}$ as well as variables $x$ with the semantics "point $p$ belongs to object $x$", the points that should be filtered out in Example 1 can be defined as those points $p$ for which the following expression is *true*:

$$Tabletop \vee Robot \vee \neg Workspace \tag{4.1}$$

Alternatively, we can follow the notational conventions of the algebra of sets and use the union, intersection, and complement operations. Then, understanding the variables as the subsets of points that belong to the respective objects, the set of points to be filtered out can be specified as follows:

$$Tabletop \cup Robot \cup \overline{Workspace} \tag{4.2}$$

Using propositional logic notation, a formalization of Example 2 is given by

$$\neg Garment \wedge \neg Person. \tag{4.3}$$

Equivalently, using set notation, we can write

$$\overline{Garment} \cap \overline{Person}. \tag{4.4}$$

In our examples, the elementary (atomic) semantic units have been *Tabletop*, *Robot*, *Workspace*, *Garment*, and *Person*. Spatially, such objects can sometimes be modeled as geometric primitives. For example, a tabletop or the workspace may be represented by a rectangular box. Otherwise, it is usually possible to approximate an object by a finite set of primitives. A simplistic representation of a robotic arm, for instance, could be a set of cylinders, each corresponding to one of the segments. The logical expressions then define the points to be filtered out on a higher semantic level. We note that using the operators $\vee$, $\wedge$, and $\neg$ (or equivalently, $\cup$, $\cap$, and the complement operator), any possible filter configuration can be expressed, i.e., the set of operators is *functionally complete*. Moreover, any particular expression can be given in a standardized form, e.g., in *Disjunctive Normal Form (DNF)*. A formula in DNF is a disjunction of conjunctive clauses (orange symbols indicate optional operators):

$$\bigvee_i \left( \bigwedge_j \neg x_{ij} \right) \tag{4.5}$$

This is equivalent to a union of intersections in set notation:

$$\bigcup_i \left( \bigcap_j \overline{x_{ij}} \right) \tag{4.6}$$

**XML-based Configuration** Filter specifications should be both easy to generate for a human experimenter and understandable to a robot. Therefore, our semantic point cloud filter can be configured in a formal yet human-readable language, namely *Extensible Markup Language (XML)*[10]. The configuration files for our examples are presented in Listings 4.1 and 4.2. Atomic variables can be declared by use of the *primitivegroup* tag. The *id* attribute gives each

[10] `https://www.w3.org/XML`

Listing 4.1: XML filter configuration corresponding to Example 1

```
<pointcloudfilter>
  <primitivegroup id="Tabletop" regex="tabletop" />
  <primitivegroup id="Robot"
      regex="rh_\w+|ra_\w+|lh_\w+|la_\w+" />
  <primitivegroup id="Workspace" regex="workspace" />
  <remove>
    <group id="Tabletop" part="inner" />
    <group id="Robot" part="inner" />
    <group id="Workspace" part="outer" />
  </remove>
</pointcloudfilter>
```

Listing 4.2: XML filter configuration corresponding to Example 2

```
<pointcloudfilter>
  <primitivegroup id="Garment" regex="garment_\w+" />
  <primitivegroup id="Person" regex="person_\w+" />
  <remove>
    <intersection>
      <group id="Garment" part="outer" />
      <group id="Person" part="outer" />
    </intersection>
  </remove>
</pointcloudfilter>
```

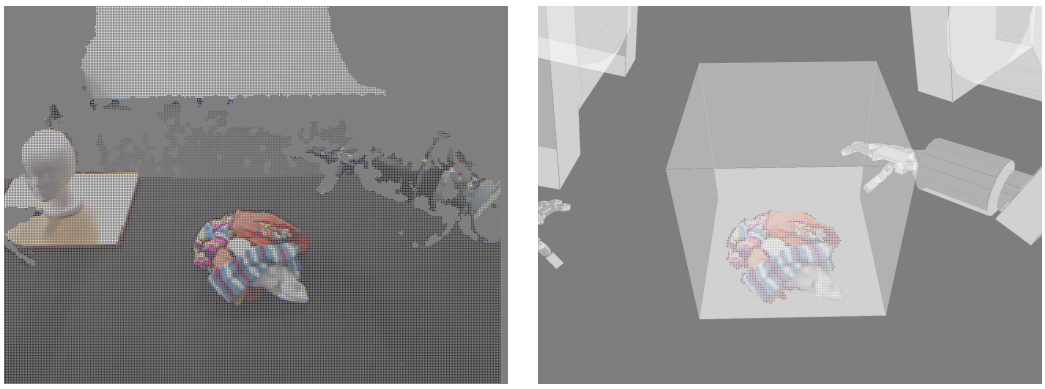
Listing 4.3: Semantic point cloud filter algorithm in pseudocode

```
for all primitives
  primitiveGroup = matchRegEx(...)
  for all points p // geometry (parallelized)
    groupMap[p][primitiveGroup] |= isInside(p, primitive)
for all formulas
  for all points p // logic (parallelized)
    actionMap[p] = evalFormula(formula, p, groupMap)
  performFilterAction(pointCloud, actionMap)
```

variable a name, and the *regex* attribute indicates, by means of a regular expression, which geometric primitives (each associated with a description string) should be grouped together to form a semantic unit. The *Tabletop* object in Listing 4.1, for instance, consists of a single "tabletop" primitive, whereas the *Robot* object combines all geometric primitives whose description strings begin with "rh_" (right hand), "ra_" (right arm), "lh_" (left hand), or "la_" (left arm). We emphasize that these primitive groups have to be disjoint (mutually exclusive), i.e., a primitive cannot be part of two groups at the same time.
Between the opening and closing *remove* tags, the points to be removed are specified in DNF, i.e., by a sequence of intersections (indicated by *intersection* tags) that are implicitly assumed to form a union. If a conjunctive clause (intersection) consists of only one literal (*group* tag), such as in Example 1, the *intersection* tag can be omitted. The *group* tags either refer to an atomic variable (*part="inner"*) or to its negation/complement (*part="outer"*).
Removing points renders *organized* point clouds *unorganized* because the structuring 2D array now contains invalid references. Therefore, rather than actually removing the points, it is also possible to specify another action such as moving the points to a distant position (using the *setpos* tag) or labeling them as invalid (using the *label* tag). Furthermore, *color* and *intensity* actions are provided for visualization, and the source depth image can be manipulated using the *filterdepthimg* tag. A general specification of the XML filter configuration can be found on the *ICL* website[11]. The geometric primitives are defined either by using the filter API included in *ICL* or through *RST*[12] messages sent via *RSB* which specify the primitive type (cube, sphere, or cylinder), the pose, the scale, and the description string.

**Parallel Implementation** In Listing 4.3, our filter implementation is summarized in pseudocode. Efficiency has been achieved by parallelizing the algorithm using *OpenCL*. Parallel algorithms can run concurrently on different processing devices such as the cores of a *Graphics Processing Unit (GPU)* and are particularly useful if many independent computations have to be performed for a large number of similar items such as the points of a point cloud. Thus, we were able to parallelize both the geometry and the logic part of our filter algorithm.
In OpenCL, code that is executed in parallel on many devices is called a *kernel*. The *isInside* kernel performing the geometric computations in our algorithm consists of two basic steps: (i) shifting and rotating $p$ according to the primitive's position and orientation (resulting in a point $p'$), and (ii)

[11] `http://www.iclcv.org/tutorials/pointcloud-filtering.html`
[12] `https://code.cor-lab.de/projects/rst`



(a) (b)

Figure 4.1: Point cloud filtering result. (a) A point cloud consisting of points that correspond to a heap of clothes, the robot, a tabletop, a styrofoam head, and some other background objects. (b) After filtering, the point cloud only contains the heap of clothes. The geometric primitives that were used for filtering are depicted in transparent gray.

checking if $p'$ is inside the non-rotated primitive centered at the origin. The result of this kernel is stored in a *group map* indicating for each point whether it belongs to a certain primitive group or not. The group map is then used in another kernel that evaluates the logical formulas specifying the points to which a particular action (e.g., removing) should be applied.

**Evaluation** Figure 4.1(a) shows the point cloud we used for evaluating the performance of our filter implementation. It includes several possibly irrelevant objects such as the robot arms and hands as well as the styrofoam head that was part of our policy learning setup in Chapter 6. In our tests, the algorithm filtered out all points outside the workspace which was specified by a cube and, depending on the experimental condition, all points belonging to the robot (according to the current joint angles and a description of the links by a set of primitives). The resulting point cloud is visualized in Figure 4.1(b).
We measured the runtime of the filter implementation following a $3 \times 2$ factorial design. For one thing, we were interested in the extent to which parallelization improved the performance of the algorithm. In particular, we tested a non-parallelized, a fully parallelized, and a partially parallelized implementation (in which only the *isInside* kernel was run in parallel). For another thing, we analyzed the effect of the amount of geometric primitives used on the algorithm's runtime. In the first condition, only one 0.5 cm $\times$ 0.5 cm $\times$

Table 4.1: Runtime of the semantic point cloud filter implementation

| runtime in [ms] | not parallelized | parallelized | only geometry parallelized |
|---|---|---|---|
| one primitive | 4.7 | 3.1 | 3.6 |
| 124 primitives | 97.7 | 8.6 | 9.2 |

0.5 cm cubic primitive representing the workspace was used for filtering. In the second condition, there were 123 additional primitives corresponding to the arms and hands of the robot.

The experimental results are shown in Table 4.1. It is not surprising that the number of primitives had a large effect on the filter performance. While a runtime of 4.7 ms in the non-parallelized implementation with a single primitive would still allow for real-time processing, 97.7 ms in the condition with 124 primitives would be a prohibitive amount of time. Parallelizing both the geometric computations and the evaluation of the logical formulas significantly reduced the runtime (to 3.1 ms in the single-primitive condition and 8.6 ms in the many-primitives condition). In the case of more than a hundred geometric primitives, this was a speed-up of one order of magnitude. It has become apparent that most of the increase in performance has been due to the parallelization of the geometry part of the algorithm. Evaluating the logical expressions sequentially for each point only slightly increased the runtime (to 3.6 ms in the single-primitive condition and 9.2 ms in the many-primitives condition).

Besides its efficient implementation in ICL and OpenCL, our algorithm has two key advantages over existing point cloud filters. First, and in contrast to camera-centric approaches[13], the pose of the depth sensor need not be known. Consequently, our technique is applicable to any type of point cloud, regardless of its source. Second, and in contrast to an otherwise similar method[14], our algorithm is not only meant for robot self filtering but, due to its functional completeness, can deal with semantically complex configurations of nested and/or overlapping objects.

[13] `http://wiki.ros.org/rgbd_self_filter`
[14] `http://wiki.ros.org/robot_self_filter`

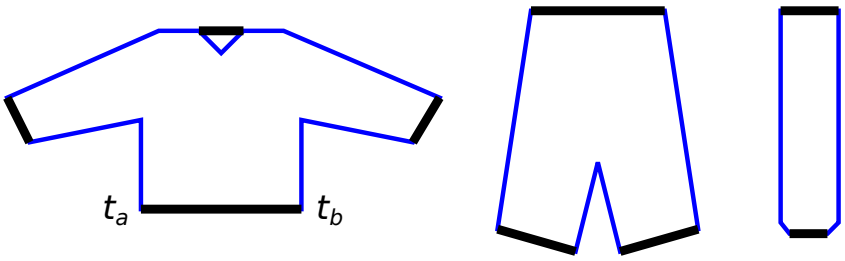



Figure 4.2: Template polygons of a sweater, a pair of pants, and a legwarmer. The line segments associated with the garment openings are depicted in black.

## 4.2 Polygon-based Boundary Component Detection

We are now ready for our first detection algorithm. In this section, we describe an approach to finding all boundary components (Section 2.2) of a piece of clothing using topological and geometric prior knowledge about different garment categories. After applying the semantic filter from Section 4.1.2, we can assume that the point cloud used is free from distracting background objects. Specifically, in the following, we assume that it only contains a single item of clothing and a support surface (the tabletop). In other words, we have reduced the problem space such that we can now straightforwardly focus on two key problems of garment recognition: (i) classifying the garment category and (ii) detecting the most relevant features, namely the openings.
However, the overall geometry of a piece of clothing can be extremely complex. Therefore, we further simplify the problem by assuming that the garment has been more or less accurately spread out on the tabletop. Then, it usually takes a polygonal shape that is characteristic of its category (Figure 4.2). This representation is related to the garment topology in that typically projections of the boundary components appear as line segments of the polygon (depicted in black) which represent the openings of the garment.

### 4.2.1 Preliminaries

It is natural to ask how realistic the assumed initial garment configuration is, and we give a twofold answer to this question. For one thing, depending on the scenario, a human can easily spread out the garment on the tabletop before the robot starts to perceive and possibly interact with it. In our evaluation of the method, this was done by the experimenter. For another thing, the preliminary

actions could also be carried out by the robot. Then, the difficulty of bringing the garment into the desired configuration depends on the previous state of the garment, again. We distinguish three possible situations, for each of which potential solutions have been discussed in the literature: (i) If the fabric is only slightly wrinkled, it may be sufficient to perform a procedure for flattening the garment surface [95, 96]. (ii) If the garment has been accurately folded over a certain axis, it can be unfolded using the approach from [94]. (iii) Otherwise, it may be possible to use one of the interactive methods for unfolding a randomly folded garment mentioned in Section 3.3.
Given the state of the art, we think that it is indeed reasonable to start from an item of clothing lying spread out on a flat surface. This allows us to encode our geometric knowledge about different garment categories as 2D polygonal templates with vertices $t_1, ..., t_n$, similar to [101] and [102]. In addition to this, we require the (topological) knowledge of how many openings a garment of a particular category has and the (geometric) knowledge of which line segments $\overline{t_a t_b}$ correspond to the openings. In the following, we will consider legwarmers, pairs of pants, and sweaters, i.e., garments with two, three, and four openings, respectively. For a more fine-grained classification, one could, of course, use many different polygonal templates, but we decided to limit the database to one prototypical polygon for each of the three mentioned categories. Thus, Figure 4.2 provides a complete description of the prior knowledge used throughout our experiments.

### 4.2.2 Algorithms

Our polygon-based boundary component detection method consists of the following four steps:

1. Hybrid foreground-background segmentation
2. Polygon matching
3. Heuristic search for openings
4. Projection onto the tabletop

**Hybrid Foreground-Background Segmentation** In theory, the semantic point cloud filter could be used not only for removing the points belonging to the robot or to irrelevant objects placed on the tabletop, but also for separating the garment points from the tabletop points (assuming that the pose of the tabletop plane is known or can be easily determined). However, in practice, there are several reasons why this segmentation method is not accurate

(a) (b) (c)

Figure 4.3: Steps of the hybrid foreground-background segmentation method. (a) Depth-registered color image. (b) Depth-based pre-segmentation result. (c) Segmentation result after applying the color-based GrabCut algorithm. Foreground pixels are depicted in dark gray, background pixels in light gray.

enough for estimating the shape of the polygon that represents the garment's 2D projection onto the support surface. First, there is always some noise in the depth data provided by the Kinect sensor. Second, the tabletop pose estimation might be imperfect. Third, and most importantly, we require the garment to lie flat on the tabletop which has the effect that the differences in depth between foreground (garment) and background (tabletop) points can be very small. Alternatively, one could employ a purely color-based segmentation approach. However, these methods tend to fail if the color ranges of the foreground and background overlap immoderately, especially if simplistic or improperly initialized color distribution models are used. Therefore, we suggest a hybrid approach to foreground-background segmentation, using both color and depth information.

The input to the algorithm is a depth-registered color image (Figure 4.3(a)) together with the corresponding point cloud. We begin by cutting the point cloud in two parts slightly above the support surface using a separating plane whose normal vector, in our case, results immediately from the point cloud calibration procedure because the table was used to define the world coordinate frame, i.e., the calibration object (Section 4.1.1) was placed on the tabletop. In other cases, one could employ any plane fitting method to estimate the pose of the support surface. The result of this pre-segmentation step is visualized in pixel space in Figure 4.3(b).

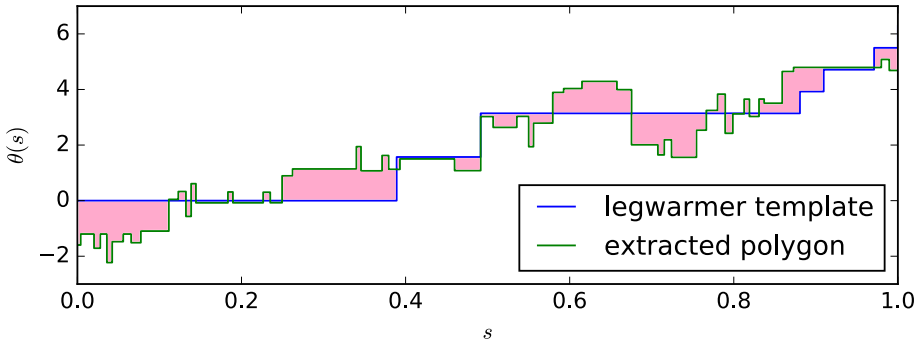
We can now assume that the set of foreground pixels only contains points that belong to the garment whereas the background pixel set contains many tabletop points and only relatively few garment points. Therefore, these sets are well-suited for learning color distribution models of the foreground and

background, respectively, and for initializing the *GrabCut* segmentation algorithm [127]. In GrabCut, the color distributions are represented by *Gaussian mixture models (GMMs)*, and the segmentation is iteratively improved such that an energy function is minimized. The energy function consists of a data term that evaluates how well the foreground and background pixels of a given segmentation fit the current color distribution models, and a smoothness term that encourages a smooth segmentation boundary. The minimization problem is solved by means of a graph cut technique [128]. We use the OpenCV implementation of the GrabCut algorithm. The final segmentation result is shown in Figure 4.3(c).

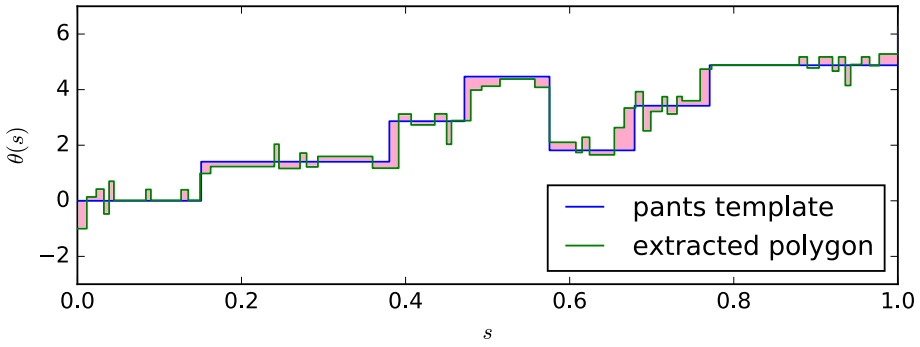
**Polygon Matching** The next step is to approximate the foreground region representing the garment by a polygon with vertices $v_1, ..., v_m$. For this, we employ the curvature-based corner detector from [129]. The polygon needs to be corrected for the camera perspective to make it commensurable with the template polygons. Therefore, the polygon approximation is not carried out in pixel space but in a planar coordinate system parallel to the support surface. To compare the extracted polygon with each of the templates, a matching algorithm is required which should output (i) the garment category (legwarmer, pair of pants, or sweater) that best fits the data and (ii) additional geometric information that can be used for finding the line segments corresponding to the garment openings. The *turning function* based method from [130] meets these requirements.
First, the extracted polygon as well as the three template polygons are scaled to have length 1. Then, we compute a turning function for each of the polygons. The turning function $\theta(s)$ of a polygon $v_1, ..., v_m$ describes how its tangent turns in the plane as a function of the (counterclockwise) arc length $s$. Specifically, $\theta(0)$ is the counterclockwise angle of the tangent at some reference point on the polygon boundary, as measured with respect to the x-axis, and $\theta(s)$ increases with left-hand turns and decreases with right-hand turns at each vertex of the polygon such that $\theta(1) = \theta(0) + 2\pi$. The $L_2$ distance between two turning functions $\theta_X$ and $\theta_Y$ is defined as follows:

$$d_2(\theta_X, \theta_Y) = ||\theta_X - \theta_Y||_2 = \left( \int_0^1 |\theta_X(s) - \theta_Y(s)|^2 ds \right)^{\frac{1}{2}} \tag{4.7}$$

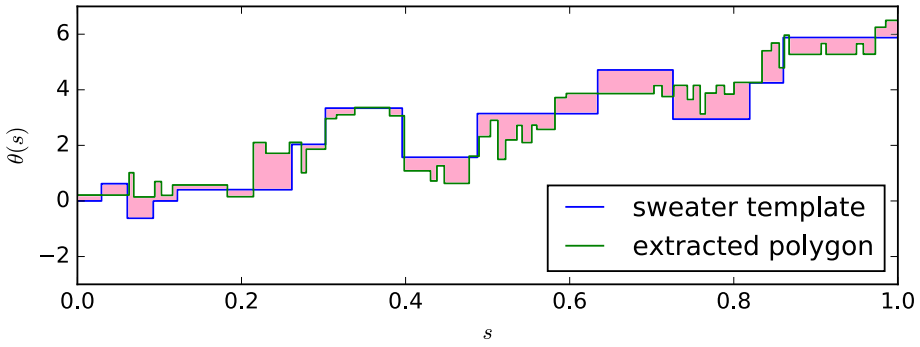
However, $d_2$ is sensitive to both rotation of and choice of reference point on the boundary of polygon $X$. Rotating the polygon corresponds to a vertical shift $\phi$ of $\theta_X$ whereas choosing the reference point differently corresponds to a horizontal shift $u$. Therefore, the metric describing the degree of shape similarity between two polygons $X$ and $Y$ is defined as the minimum of $d_2$




(a)




(b)




(c)

Figure 4.4: Turning function of the extracted polygon (green) matched against each of the three templates (blue). (a) Legwarmer template. (b) Pants template. (c) Sweater template. The light pink areas correspond to the turning function distances.

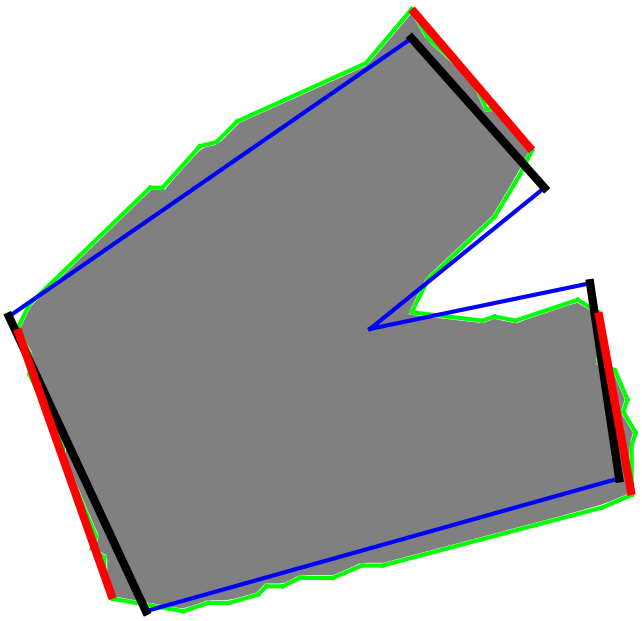


Figure 4.5: Result of the polygon matching and the heuristic search for garment openings. The extracted polygon and the best-matching template are shown in green and blue, respectively. The optimal segments (representing the openings) are depicted as red lines, the corresponding template segments as black lines.

over all $\phi$ and $u$ so as to be independent of rotation and choice of reference point:

$$m(X,Y) = \min_{\phi,u} \left( \int_0^1 |\theta_X(s+u) - \theta_Y(s) + \phi|^2 ds \right)^{\frac{1}{2}} \tag{4.8}$$

In Figure 4.4, the template turning functions are depicted in blue, and the turning function of the polygon to be matched (which was extracted from the example images shown in Figure 4.3) is depicted in green. The three green graphs all show the same turning function, but shifted both vertically and horizontally to match the respective template functions optimally. The turning function distances are visualized in light pink. It is apparent that the light pink area in Figure 4.4(b) is much smaller than those in Figures 4.4(a) and 4.4(c), meaning that the pair of pants is classified correctly. Specifically, $m$ is 0.89 for the legwarmer, 0.75 for the sweater, and only 0.43 for the pair of pants. Both the horizontal shift $u$ and the vertical shift $\phi$ of the best match are stored to be used in the next step.

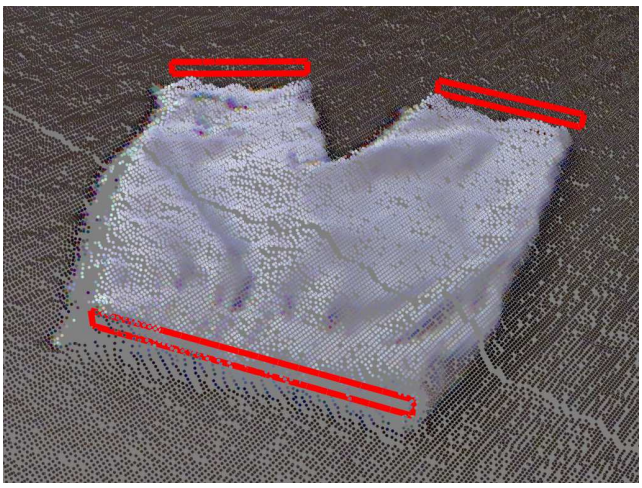


Figure 4.6: The input point cloud and the result of the boundary component projection (red) onto the support surface (tabletop).

**Heuristic Search for Openings** The vertical shift $\phi$ corresponds to the rotation that was necessary to achieve the optimal match. Therefore, knowing $\phi$ makes it possible to lay the best-matching template on top of the extracted polygon in such a way that the overall orientations, the arc lengths, and the centroids match. In Figure 4.5, the extracted polygon is shown in green and the matching pants template is depicted in blue.
Moreover, using the horizontal shift $u$ (and assuming all polygons to be scaled to length 1), we are able to define an arc distance between a template vertex $t_i$ and a vertex $v_j$ of the extracted polygon. For this purpose, we consider two corresponding reference points $t_r$ and $v_R$, where one reference point can be chosen arbitrarily (e.g., $t_r = t_0$) and the other one is determined by the value of $u$ (e.g., obtain $v_R$ by starting at $v_0$ and going $u$ units along the polygon boundary). Then, using the counterclockwise arc lengths $L(\widehat{t_r t_i})$ and $L(\widehat{v_R v_j})$, we define the arc distance as follows:

$$d_{arc}(t_i, v_j) = \min\left(|L(\widehat{t_r t_i}) - L(\widehat{v_R v_j})|, 1 - |L(\widehat{t_r t_i}) - L(\widehat{v_R v_j})|\right) \tag{4.9}$$

Identifying the 2D projection of a garment opening is the problem of finding the vertices $v_A$ and $v_B$ of the extracted polygon that correspond to $t_a$ and $t_b$ in the best-matching template. First, we determine all approximately linear segments $\overline{v_i v_j}$. By this, we mean that the distances of intermediate vertices to the connecting line do not exceed a threshold. Strictly speaking, we define two thresholds, one for outlying vertices (i.e., vertices to the right of the connecting line in the counterclockwise case) and a higher one for inlying vertices, relaxing

the linearity condition for concave segments such as the neckline of a sweater. Then, we minimize a heuristic cost term $F$ over all $\overline{v_i v_j}$ to find $\overline{v_A v_B}$:

$$F = \alpha_1 F_{arc} + \alpha_2 F_{dist} + \alpha_3 F_{orient} + \alpha_4 F_{length}, \tag{4.10}$$

where

$$F_{arc} = d_{arc}(t_a, v_i) + d_{arc}(t_b, v_j), \tag{4.11}$$

$$F_{dist} = ||t_a - v_i|| + ||t_b - v_j||, \tag{4.12}$$

$$F_{orient} = \arccos\left(\frac{(t_b - t_a) \cdot (v_j - v_i)}{||t_b - t_a|| \cdot ||v_j - v_i||}\right), \tag{4.13}$$

and

$$F_{length} = \left|1 - \frac{||v_j - v_i||}{||t_b - t_a||}\right|. \tag{4.14}$$

While $F_{arc}$ makes use of the arc length parameterizations of the polygons, the other three terms operate in Euclidean space. Specifically, $F_{dist}$ corresponds to the Euclidean distance between the line segments, $F_{orient}$ is the angle between $\overline{t_a t_b}$ and $\overline{v_i v_j}$, and $F_{length}$ relates to their difference in length. The coefficients $\alpha_1$, $\alpha_2$, $\alpha_3$, and $\alpha_4$ control the relative influence of the terms. The segments $\overline{v_A v_B}$ that minimize $F$ are shown as red lines in Figure 4.5, and the corresponding template segments are depicted in black.

**Projection onto the Tabletop** Finally, the optimal line segments $\overline{v_A v_B}$ are back-projected to the 3D world. Since the polygon approximation was performed in a coordinate system parallel to the tabletop, we can simply shift the vertices $v_A$ and $v_B$ of the original (i.e., unscaled) polygon along the normal vector of the support surface and connect them to obtain the contact line between a garment opening and the tabletop. This contact line is then completed to a rectangular boundary component model of height 1 cm as visualized in Figure 4.6.

### 4.2.3 Evaluation

In our experiments, we used a test set consisting of six different child garments from three categories (i.e., two leg warmers, two pairs of pants, and two sweaters) as shown in Figure 4.7. We performed twelve runs of the first three steps of the method described in Section 4.2.2 (omitting the conversion to a

Figure 4.7: Test set used for evaluating the polygon-based boundary component detection method.

(a) (b)

Figure 4.8: Two trials from the polygon-based detection experiment. Ground truth segments provided by a human participant are shown in yellow. (a) A leg warmer in the black background condition. (b) A sweater in the multicolored background condition.

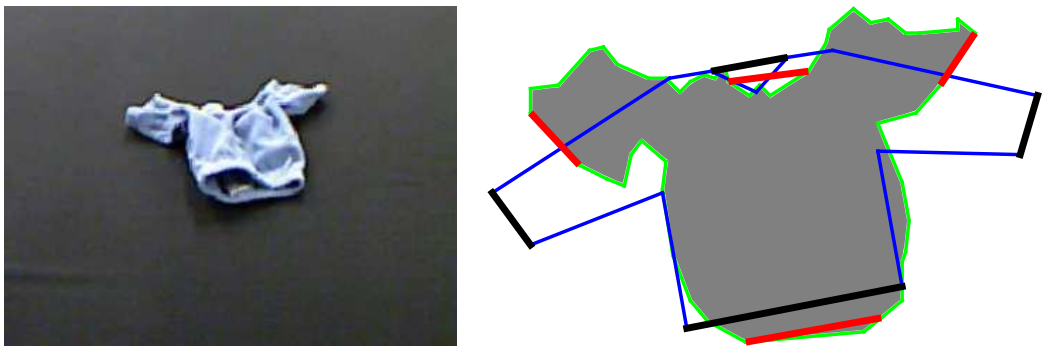
Table 4.2: Polygon-based boundary component detection accuracy

| error with respect to ground truth | black background | | multicolored background | |
|---|---|---|---|---|
| | mean | SD | mean | SD |
| position error | | | | |
| in pixels (image frame) | 3.0 | 1.4 | 7.9 | 11.2 |
| in mm (world frame) | 7.4 | 2.9 | 18.9 | 23.2 |
| length error | | | | |
| in pixels (image frame) | 4.8 | 4.5 | 4.1 | 4.2 |
| in mm (world frame) | 12.2 | 10.5 | 10.6 | 10.5 |
| orientation error | | | | |
| in degrees (image frame) | 3.2 | 2.4 | 5.6 | 5.3 |
| in degrees (world frame) | 3.4 | 2.4 | 6.1 | 5.3 |

rectangular model). The coefficients of the cost term $F$ were chosen as follows: $\alpha_1 = 1.0$, $\alpha_2 = \alpha_3 = \alpha_4 = 0.5$. Each item from the test set was spread out once on a black tabletop and once on a multicolored table cloth.

The first result is that the algorithm classified the garment category correctly in all trials. In addition to this, we testet the accuracy of the method by comparing the results with the ground truth given by a human subject. The participant was presented with the same twelve depth-registered color images and was asked to mark the openings of the garments (Figure 4.8). Afterwards, we measured the differences in position, length, and orientation between the algorithm-generated segments and ground truth (Table 4.2). In the black background condition, errors in position (7.4 mm), length (12.2 mm), and orientation (3.4 degrees) were small. While length errors did not differ much between the conditions, both position and orientation accuracy suffered from the poorer segmentation results in the multicolored background condition.

In the quantitative experiments, we assumed the items of clothing to lie more or less flat on the tabletop. But we were also interested in how stretching, crumpling, and folding the garments influence the results of the algorithm. Therefore, we performed some additional qualitative tests in which, in the first instance, we deformed the garments only slightly. We found that the algorithm is quite forgiving of moderate deformations as far as classification is concerned (at least in the three-class case considered), but the resulting line segments were sometimes inaccurate because the position and orientation assumptions given by the templates were violated. Finally, we tested the method with more heavily deformed garments which lead to misclassifications such as the



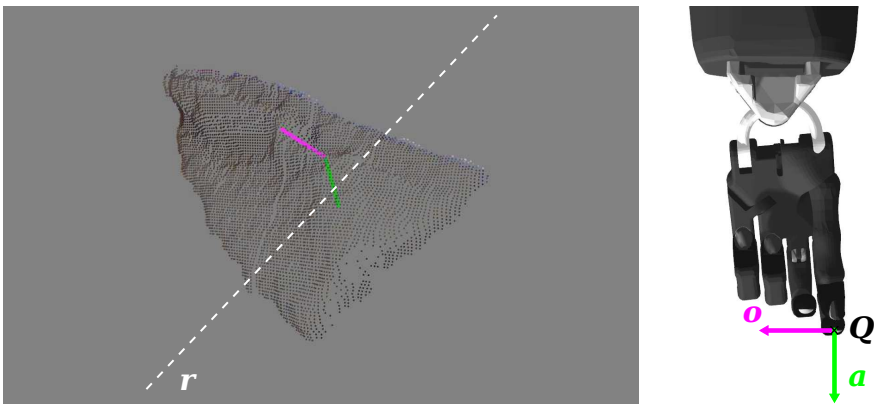
(a) (b)

Figure 4.9: Failed polygon matching trial. (a) Depth-registered color image of a deformed pair of pants. (b) The pair of pants is misclassified as a sweater.

one illustrated in Figure 4.9. However, this was an expected result, considering the fact that the polygon matching algorithm only operates in 2D.

## 4.3 Interactive Boundary Component and Optimal Grasp Pose Detection

Although we were able to show the effectiveness of the polygon-based boundary component detection method, there are some obstacles to its practical application in robotic interaction with clothes. First, it may not always be trivial to meet the requirement that the garment should be spread out flat. Second, the areas of the openings of a flattened garment are, of course, very small. Consequently, there is not much space for reaching into the openings with one or more fingers. Third, while the positions, orientations, and lengths of the openings are detected quite well, the boundary curves are only approximated by rectangular models. Nevertheless, this simplistic representation of the boundary components is well-suited for initializing a dynamic model, as will be seen in Chapter 5.
However, it is also possible to rely on static images, but to make use of interactive perception. By this, we mean that the robot changes the configuration of the environment by lifting the garment from the support surface (Section 4.3.1) so as to facilitate perception. The method described in Section 4.3.2 then uses a graph representation for boundary component detection, which allows us to define a grasp pose around the boundary of a garment opening (Section 4.3.3). The main disadvantage of this technique is that it can typi-




(a) (b)

Figure 4.10: Initial grasp configuration for interactive boundary component detection. The approach and orientation vectors specifying the hand frame during grasping are depicted in green and pink, respectively. (a) Point cloud of a knit cap and extracted grasp pose. (b) Closed hand posture.

cally only detect one opening. Therefore, it is primarily intended to be used with single-opening garments (such as socks or knit caps) or to model only the most relevant boundary component such as $b_0$ in robot-assisted dressing (Section 2.3).

### 4.3.1 Initial Grasping

One of the key properties of any interactive perception scheme is a more or less targeted interaction with the environment that precedes the main process of perception (e.g., boundary component detection). The initial robotic action is of an exploratory nature, i.e., it is often rather simple and sometimes affected by a certain amount of randomness. Nevertheless, it must serve a clearly defined purpose, namely to transform the environment into a state in which the perception task is simplified or solvable in the first place. In our case, that means making the garment opening visible to the camera. To this end, the robot grasps the garment using one of its hands and slightly lifts it from the tabletop. Several sophisticated approaches to finding an optimal pick-up grasp pose for a piece of clothing being placed on top of a flat surface have been suggested in the literature (Section 3.3). But these techniques cannot guarantee that the garment opening is made visible. Therefore, we present

a new heuristic method exploiting gravity and the approximate shape of the garment considered.

We distinguish two different concepts that, taken together, determine the configuration of the grasping hand. The first concept is that of a *grasp pose*. By this, we mean the grasp position $Q$ along with the orientation of the hand frame relative to the world frame. An intuitive way to specify the hand frame orientation during grasping is by two orthogonal unit vectors $\vec{a}$ and $\vec{o}$. The third coordinate axis $\vec{n}$ is then given implicitly by the cross product. The *approach vector* $\vec{a}$ (depicted in green in Figure 4.10) defines the direction along which the hand should approach the object to be grasped. The *orientation vector* $\vec{o}$ (pink) specifies how the hand is oriented in the plane orthogonal to $\vec{a}$. To compute the rotation between the world frame and the hand frame, $\vec{a}$, $\vec{o}$, and $\vec{n}$ can be considered the columns of a rotation matrix which can be easily converted to an Euler angle or unit quaternion representation, if needed. The second concept is that of a *hand posture* which is defined by the joint angles of the wrist and the fingers. Specifically, the procedure of picking up a garment will be characterized by three hand configurations (i.e., three grasp poses and three hand postures): a pre-grasp configuration (before closing the hand), a grasp configuration (of the closed hand), and a post-grasp configuration (the target configuration of the hand holding the garment).

We now describe a simple heuristic method for grasping a certain class of knit caps. While this particular technique is not applicable to other types of clothing, we think that similar heuristics can be used to find suitable initial grasp configurations for differently shaped garments. The main idea is not to apply a method that tries to be exact such as the polygon-based approach from Section 4.2, but to make use of approximate measures that are tolerant of minor garment deformations. We make the assumption that when dropping a knit cap (of the type considered in our experiments) on a flat surface, it retains an elongated shape, even if it is slightly crumpled. Under this condition, our method extracts a grasp pose from point cloud data, which is then also used for deriving the pre- and post-grasp configurations.

We assume that the knit cap has been isolated in the input point cloud using the filter from Section 4.1.2. Of course, the garment-tabletop segmentation may be imperfect again. But for estimating an initial grasp pose, this is less of a problem because the method is not based on the exact shape of the garment but only on the statistical distribution of the points. We perform a 2-dimensional *principal component analysis (PCA)* of the knit cap points, omitting the vertical coordinates. Let $\vec{\mu}$ be the mean of the 2D points, let $\lambda_1$, $\lambda_2$ be the eigenvalues of the covariance matrix, and let $\vec{v_1}$, $\vec{v_2}$ be the corresponding eigenvectors. Now, $\vec{v_1}$ points approximately in the direction of either the

opening or the opposing closed end of the knit cap. For the sake of simplicity, we leave this binary distinction to the user and from now on assume that $\vec{v_1}$ is directed roughly toward the garment opening. Then, a line $\vec{r}$ (visualized as white dashed line in Figure 4.10(a)) can be defined as

$$\vec{r} = \vec{\mu} + \frac{1}{2}\sqrt{\lambda_1}\vec{v_1} + t \cdot \vec{v_2} \tag{4.15}$$

for $t \in \mathbb{R}$. The highest point within a distance of 1 cm to this line is selected as the grasp position $Q$. The approach direction is straight downward (i.e., $\vec{a}$ is orthogonal to the tabletop plane) and the orientation vector $\vec{o}$ is set to $-\vec{v_1}$ (Figure 4.10(a)). In order to cover the whole range of possible angles, the grasping hand is chosen depending on whether the orientation vector points more to the left or to the right. The hand posture during grasping is shown in Figure 4.10(b). It can be seen that the thumb and the forefinger are almost in contact.
In the pre-grasp hand posture, by contrast, there is some space between the thumb and both the forefinger and the middle finger. The pre-grasp position is set to $Q - d_{pre} \cdot \vec{a}$, where $d_{pre} = 5$ cm, i.e., the finger tips are positioned 5 cm above the garment. The post-grasp pose is predefined such that, assuming a successful grasp at a point in the knit cap's front area, the garment hangs in a good configuration to be observed by the depth camera. Optimally, the orientation vector would be orthogonal to the image plane. However, this pose is not reachable by the robot arm due to its joint limits. Therefore, the post-grasp hand posture is defined to be different from the grasp posture in that the ring finger and the little finger are stretched out so as to push the hanging knit cap to a pose in which the opening is closer to parallel to the image plane.

### 4.3.2 Algorithms

In the following, we assume that the garment has been picked up from the tabletop successfully and hangs freely under gravity in such a way that the opening (or one of the openings) is visible to the robot's camera. We present an approach to finding the closed contour that represents the boundary component corresponding to the opening. We first show how to create a graph representation of thinned edges which is then used to extract the boundary component. Specifically, our graph-based boundary component detection method consists of the following steps:

1. Edge detection
2. Skeletonization

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

Figure 4.11: Intermediate results of the graph-based boundary component detection method. (a) Color image of a knit cap held by a human hand. (b) Depth image visualized as a heat map. (c) Edge image. (d) Skeletonized edge image. (e) Skeleton graph and selected cycle (depicted in red).

3. Skeleton graph creation
4. Finding simple cycles
5. Selecting simple cycles
6. Back-projection to 3D

**Edge Detection** Visually, the boundary components of garments appear as edges that form closed curves. In the ideal case, these edges can be detected in color or grayscale images using, e.g., the well-known *Canny edge detector* [131]. However, detectors based on intensity differences in the image tend to produce false positives caused by prints and patterns on the fabric and miss relevant edges in the case of homogeneously colored garments (such as the knit cap in Figure 4.11(a)). Therefore, we rely on depth images (Figure 4.11(b)) along with the corresponding point clouds, and employ a surface normal based edge detection method. We use the highly optimized implementation from Ückermann et al. [132] which can be configured to meet our requirements.
The first step is to smoothen the input depth image both temporally (by averaging over a number of successive frames) and spatially (by means of a median filter). This noise reduction step is crucial for the subsequent parts of the algorithm to be stable. Then, for each pixel, a surface normal is estimated from the plane spanned by three neighboring points. Finally, the scalar products with the normals in an 8-neighborhood are averaged before binarizing the obtained angle image employing a threshold (which is set rather low to make sure that no relevant edge is missed). The edge detection result is shown in Figure 4.11(c).

**Skeletonization** The edge points in the binary image created by the surface normal based detector often form complex shapes. In particular, the edges may be fringed and have varying thickness. In other words, the image contains overly detailed (yet sometimes inaccurate) geometric information. Therefore, the next step is to thin out the edges in such a way that only the topological and the most relevant geometric properties are preserved. For a survey of thinning (skeletonization) methodologies covering sequential, parallel, as well as non-iterative techniques, the reader is referred to [133].
Our skeletonization procedure is based on Zhang and Suen's parallel algorithm described in [134] which we reimplemented in ICL using OpenCL for parallelization. The idea of the algorithm is to iteratively remove boundary points preserving the end points and pixel connectivity. To this end, two subiterations are conducted. Southeast points are deleted in a first pass before deleting northwest points in a second pass. The result is an 8-connected thinned version of the edge image (Figure 4.11(d)).

**Skeleton Graph Creation** The next algorithm converts the skeletonized edge image to a graph representation. This is done in a two-step process. First, the junction points, i.e., the points with three or more neighbors in the image, are inserted as vertices into the graph. In a second step, we find the branches that link the junction points in the image. To this end, a new branch is created for each point adjacent to a junction point. Then, we follow each branch (consisting of points with two neighbors) until we reach another junction point or an end point (i.e., a point with exactly one neighbor). If another junction point is found (and no edge representing the same branch exists in the graph), the branch is inserted into the graph as edge linking the two vertices. Otherwise, the branch is discarded because free ends cannot be part of a simple cycle (see below).
Up to this point, the algorithm misses components without any junction point. Therefore, while there are still unvisited points, the branch following procedure is started again at an arbitrary point. This results in branches with two end points (which we discard) or subgraphs with a single vertex and a loop edge. Strictly speaking, the overall result of the algorithm (Figure 4.11(e)) is in general not a graph but a multigraph, i.e., two vertices may be linked by multiple edges which is forbidden in conventional graphs.

**Finding Simple Cycles** It is reasonable to assume that the visible boundary component of the garment is represented as a cycle in the skeleton graph. The following definitions are required for a graph-theoretic specification of the cycle finding algorithm:

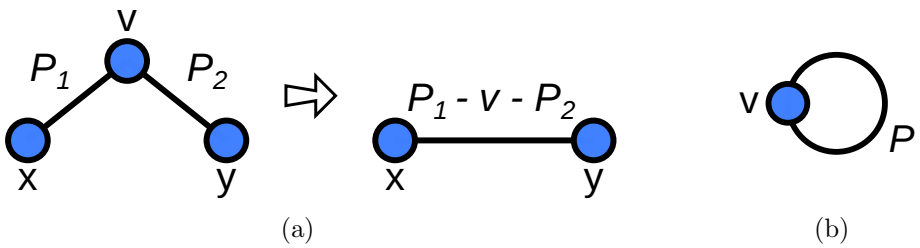



Figure 4.12: Basic idea of the path graph reduction algorithm from [135]. (a) Removing vertex $v$ and adding a new edge concatenating the paths $P_1$ and $P_2$ of two adjacent edges. (b) A simple cycle has been found if there is a loop in the path.

*A walk* is a sequence of vertices where any two consecutive vertices are linked by an edge.

*A cycle* is a walk through the graph starting and ending at the same vertex.

*A simple cycle* is a walk that does not contain any repetitions of edges or vertices, except the starting and ending vertex being the same.

*A cycle chord* is an edge that is not part of the cycle but connects vertices which are part of the cycle.

*A chordless cycle* is a cycle without any chords.

There are efficient methods for detecting all chordless simple cycles in a graph. However, the boundary component cycle is not necessarily chordless. For example, the red cycle in Figure 4.11(e) has a chord resulting from a wrinkle in the inner part of the knit cap. Consequently, all simple cycles in the graph must be detected because they are possible boundary component candidates. For this purpose, a path graph reduction method is applied. We reimplemented Hanser et al.'s algorithm [135] which returns the simple cycles as a side product of iteratively reducing the graph. The procedure is as follows: While there are vertices in the graph, the vertex $v$ of minimum degree is selected. For each pair of edges $x - P_1 - v$ and $v - P_2 - y$ with disjoint paths $P_1$ and $P_2$, a new edge $x - P_1 - v - P_2 - y$ is created (Figure 4.12(a)). If an edge's starting vertex is the same as its ending vertex ($v - P - v$), a simple cycle has been detected (Figure 4.12(b)). Then, vertex $v$ and all its edges are removed from the graph and the next vertex is selected. The algorithm terminates when there are no vertices left in the graph.
We note that, in theory, the maximum number of simple cycles in a graph (and hence the runtime of the algorithm) increases exponentially with the number

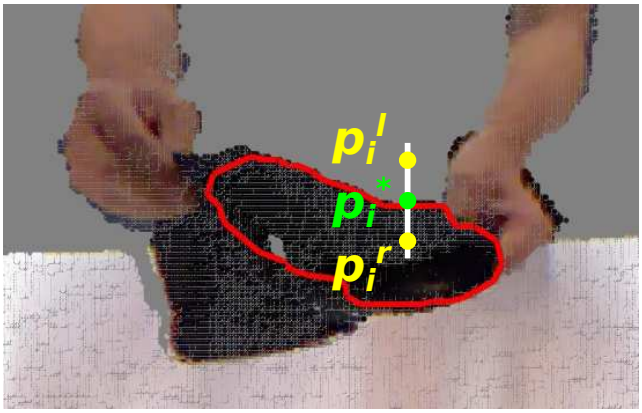



Figure 4.13: Intuition of the convexity criterion used for evaluating simple cycles: Boundary points stick out compared to points to the left and to the right of the edge.

of edges. Therefore, the search space should be reduced before applying the cycle finding algorithm. In our experiments, we only considered a region of interest around the estimated garment position, which, in the interactive perception scenario, was easily derived from the post-grasp pose of the robot hand. Another strategy for reducing the number of edges in the graph is described below.

**Selecting Simple Cycles** The simple cycles are evaluated by how well they meet a set of criteria. Here, we make use of our prior knowledge about the garment considered. For example, it is known that a knit cap has one boundary component of a certain (approximate) length. We remark that one could come up with a wide variety of criteria (e.g., based on color information). However, in our experiments, we only applied a convexity criterion in addition to the length condition. The basic idea of the convexity criterion is visualized in Figure 4.13. First, we find the pixel $p_i^*$ with the lowest depth value in a small range around each edge point $p_i$, accounting for minor deviations from the real edge caused by skeletonization. Then, after estimating the orthogonal vector to the edge (in the image plane) using $p_{i-1}$ and $p_{i+1}$, we compare the depth value of $p_i^*$ with those of two pixels $p_i^l$ and $p_i^r$ to the left and to the right of the edge. The edge point $p_i$ fulfills the criterion if it is closer to the depth camera than $p_i^l$ and $p_i^r$.
The cycle with the highest ratio of edge points satisfying the convexity criterion is selected from all simple cycles that fulfill the length criterion (if the ratio is higher than a threshold). To circumvent the exponential growth problem in connection with simple cycles, we also apply the convexity criterion to all edges

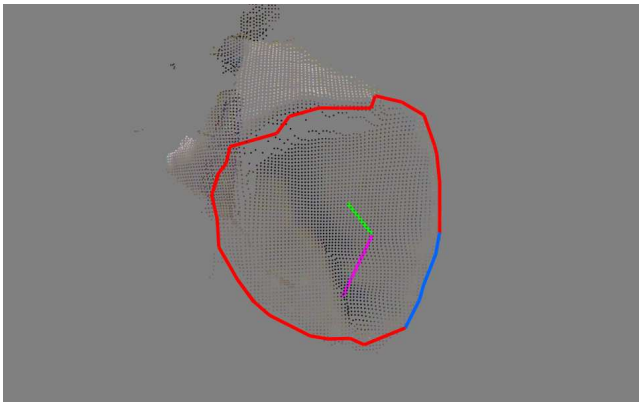


Figure 4.14: Grasp pose around the boundary of a knit cap opening. The detected boundary component is shown in red, the optimized grasp segment in blue, and the derived approach and orientation vectors in green and pink, respectively.

above a certain length before running the cycle finding algorithm, removing the edges that match the criterion worse than a threshold.

**Back-projection to 3D** Finally, the winning simple cycle is converted to a 3D boundary component model as follows: First, we approximate the cycle by a closed polygonal chain using the corner detector from [129] and, again, find the pixel $p_i^*$ with the lowest depth value in a small range around each polygon corner $p_i$. Then, we intersect the view ray of the depth camera at $p_i$ with a plane parallel to the image plane and going through the point corresponding to $p_i^*$ in the organized point cloud, adding the intersection point to the boundary component model.

### 4.3.3 The Boundary Grasp

In Section 2.1.2, we have shown that the openings provide a meaningful link between the appearance of a piece of clothing and a number of manual interaction possibilities with it (affordances). Using the example of getting undressed, we have demonstrated how such interaction primitives are applied by humans in everyday tasks. In this example, targeted grasps around the boundaries of the garment openings played a particularly important part. Furthermore, it has become apparent that boundary grasps are also needed in robot-assisted dressing and similar robotic tasks. Therefore, we will now describe a method for determining a suitable grasp configuration using the boundary component model obtained from the interactive detection procedure.

We think that good boundary segments to grasp around (such as the one depicted in blue in Figure 4.14) should satisfy the following criteria:

*Low curvature:* Segments with high curvature are likely to be corners or creased parts of the boundary. By contrast, low curvature segments usually provide more space for reaching into the opening and hence should be preferred.

*Stability:* Perceptual reliability should be taken into account when specifying grasp configurations for a robot hand. Therefore, optimal segment hypotheses that are stable over time are preferred to unstable hypotheses.

*Reachability:* Another important optimization criterion is grasp comfort. For example, the left hand should not try to grasp around a segment on the right side of the boundary and vice versa. Therefore, we prefer segments that match a desired orientation.

*Good segment length:* Optimal segments should be of a certain length corresponding to the breadth of the fingers that attempt to reach into the garment opening (plus some spacing).

Given a boundary component represented by a sequence of 3D points $p_1, ..., p_N$ in clockwise order, our optimizer iterates through all segments $p_n, ..., p_{n+l}$ of length $L \approx 5$ cm, where $L = \sum_{i=n}^{n+l-1} ||p_i - p_{i+1}||$, and finds the segment which minimizes an overall cost term $E$. The procedure is repeated for several consecutive frames. $E$ is a weighted sum of three terms:

$$E = \alpha_1 E_{curv} + \alpha_2 E_{dist} + \alpha_3 E_{orient}, \tag{4.16}$$

where the coefficients $\alpha_1$, $\alpha_2$, and $\alpha_3$ control the relative influence of the terms. $Curvature$ is approximated as follows:

$$E_{curv} = \frac{L}{||p_n - p_{n+l}||} \tag{4.17}$$

$E_{dist}$ penalizes sudden changes of position between frames and thus enforces *stability*:

$$E_{dist} = \frac{1}{2L}\big(||p_n(t) - p_n(t-1)|| + ||p_{n+l}(t) - p_{n+l}(t-1)||\big) \tag{4.18}$$

To enforce *reachability*, $E_{orient}$ is defined as follows:

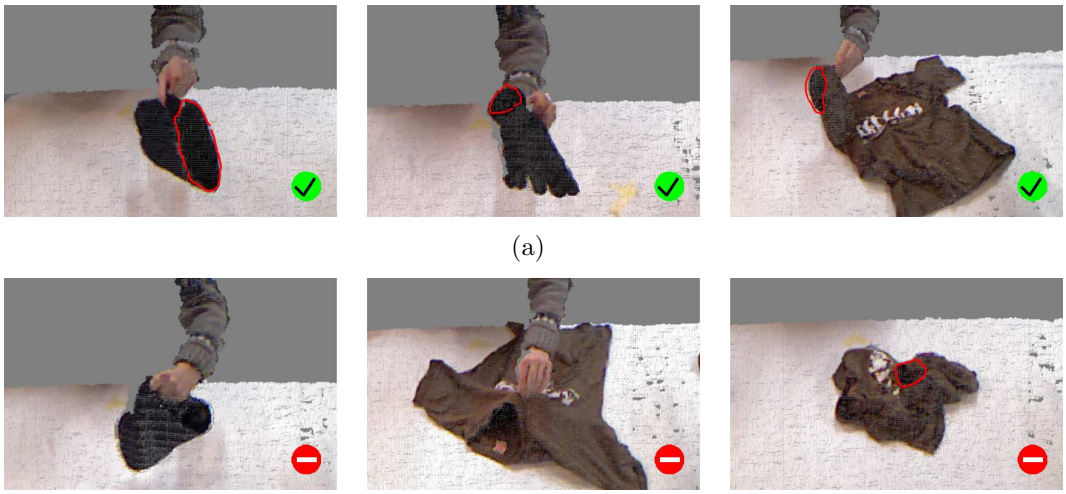
$$E_{orient} = \frac{p_n - p_{n+l}}{||p_n - p_{n+l}||} \cdot \vec{v}_{orient}, \tag{4.19}$$

where $\vec{v}_{orient} = (0, 0, 1)^T$ if the left hand is used, and $\vec{v}_{orient} = (0, 0, -1)^T$ if the right hand is used.
In the following, we describe how the boundary grasp pose is specified based on the optimal segment: The pre-grasp position is set to $Q_{pre} = \frac{p_n + p_{n+l}}{2}$ and the grasp position is defined as $Q = Q_{pre} + d_{grasp} \cdot \vec{a}$, where $d_{grasp} = 7$ cm. The approach vector $\vec{a}$ is set to the normal direction of a plane that we fit to the boundary points using a planar RANSAC algorithm [136]. We limit the approach vector to deviate from the tabletop plane by at most 20° in order to obtain a relaxed elbow pose. For the orientation vector, $p_n$ and $p_{n+l}$ are projected onto the RANSAC plane, resulting in two points $p_n^*$ and $p_{n+l}^*$. Then, we set $\vec{o} = \frac{p_n^* - p_{n+l}^*}{||p_n^* - p_{n+l}^*||}$ if the left hand is used, and $\vec{o} = \frac{p_{n+l}^* - p_n^*}{||p_{n+l}^* - p_n^*||}$ if the right hand is used. The hand postures are similar to those in Section 4.3.1. Thus, if the boundary grasp is successful, the thumb reaches into the opening of the garment while the other fingers grasp it from the outside.

### 4.3.4 Evaluation

**Boundary Component Detection** In a first series of experiments, we evaluated the graph-based boundary component detection method qualitatively in an isolated manner. To this end, a human experimenter held different items of clothing in his hand in such a way that one opening was visible to the camera. Boundary component detection was successful for garments of various types, e.g., a knit cap, a glove, and a T-shirt, as shown in Figure 4.15(a). The shortcomings of our method can be seen in Figure 4.15(b). First, we found that the algorithm was not robust to occlusion, e.g., by the human hand, because the occluding object was likely to disrupt the cycle in the skeleton graph. Second, detection failed when parts of the boundary lay flat on the tabletop such that no edge was found by the surface normal based edge detector. Third, dents in the fabric which geometrically resembled a boundary component led to false positives.

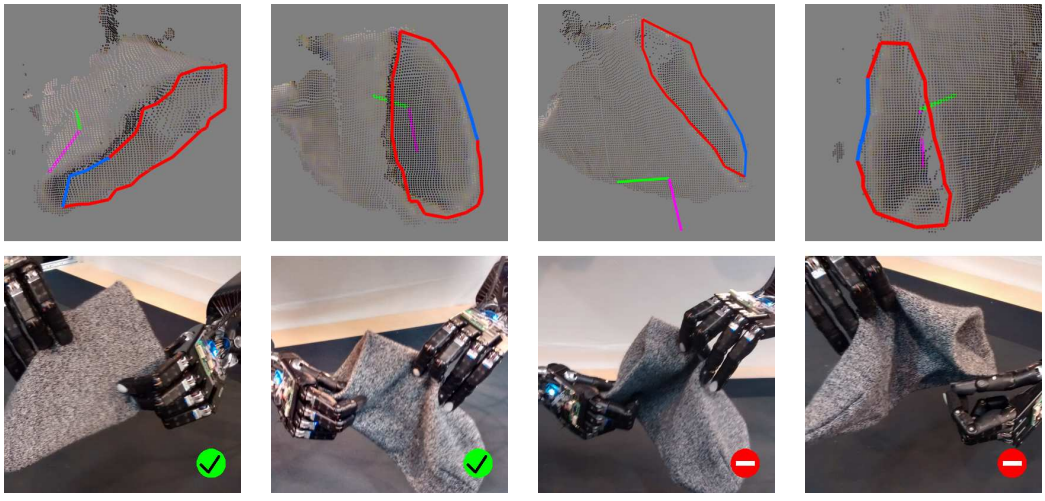
**Interactive Detection and Grasping** In a second series of tests, we evaluated the performance of the robot in a complete interactive perception and regrasping experiment with a knit cap, i.e., with a single-opening garment. The robotic setup used was the one described in Section 1.2. We conducted twelve trials in which a human experimenter held the knit cap roughly in the middle before dropping it from a height of about 50 cm above the tabletop. The robot's task was to pick up the knit cap, detect the opening, and regrasp it around a suitable segment of the boundary component. We aborted the trial if any of the subtasks failed.



(a)

(b)

Figure 4.15: Qualitative evaluation of the graph-based boundary component detection algorithm. (a) Three successful trials. (b) Three failed trials (two false negatives and one false positive).



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

Figure 4.16: Four trials from the interactive boundary component detection and grasping experiment. (a+b) Two examples of successful runs. (c+d) Two regrasping failures.

Table 4.3: Interactive detection and grasping results

| success rate | initial grasp | detection | boundary grasp |
|---|---|---|---|
| trials | 11/12 | 10/11 | 8/10 |
| percentage | 91.7 | 90.9 | 80.0 |

The robot successfully performed the initial pick-up grasp in eleven out of twelve runs. In one single case, the knit cap slipped out between the fingers. Boundary component detection was successful in ten out of eleven remaining trials. The one detection failure was due to the knit cap being turned down in such a way that the opening was not visible to the depth sensor. The robot succeeded in regrasping the knit cap around its boundary in eight out of ten runs (e.g., Figures 4.16(a) and 4.16(b)). One failure was caused by trying to grasp the garment too close to a boundary corner and unintentionally squeezing the opening (Figure 4.16(c)). The second failure occurred because the hand folded the knit cap to the side when trying to grasp it and hence missed the opening (Figure 4.16(d)). The overall success rate of the whole procedure was 66.7 percent. The results are summarized in Table 4.3.

### 4.3.5 Application

**The Coat-check Scenario** To demonstrate the usefulness of our interactive perception and regrasping approach, we created the so-called robotic coat-check scenario. In this scenario, we have shown how grasping based on a boundary component model can be exploited for implementing another common interaction primitive with clothes, namely *hanging up* a garment. Specifically, our robot performed a sequence of eight actions in order to change the configuration of a knit cap from lying on the table to hanging on a hat-stand. While the first three steps have been discussed in detail, we basically regard the (more or less preprogrammed) subsequent actions as an application of the boundary grasp. In the following, we describe the overall procedure (Figure 4.17):

1. *Picking up:* The robot picks up the knit cap from the tabletop using a heuristic-based initial grasp configuration.

2. *Visually observing:* One robot hand holds the knit cap in such a way that the boundary can be detected.

3. *Regrasping around the boundary:* Having determined a good boundary grasp pose, the robot regrasps the knit cap with the other hand.

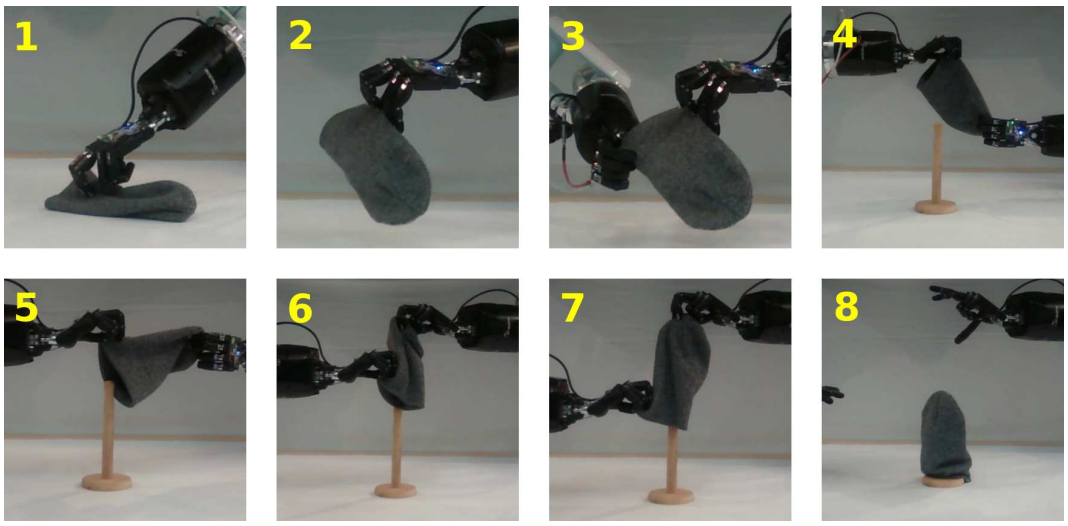



Figure 4.17: The coat-check task (hanging up a knit cap on a hat-stand) divided into a sequence of eight actions.

4. *Regrasping at the tail:* The robot performs another regrasp at a point near the closed end of the knit cap.

5. *Aligning:* The top of the hat-stand is detected in the point cloud and the knit cap is aligned accordingly.

6. *Lifting up:* One hand lifts the tail of the knit cap above the hat-stand.

7. *Pulling down:* The other hand pulls the knit cap over the hat-stand.

8. *Releasing:* The robot drops the knit cap.

**The Christmas Elf Scenario** In another demonstration scenario, we tried to bridge the gap between science and art with a christmas video that was made in collaboration with several members of our group. In the video[15], a so-called robot christmas elf explains its work (putting candy into a stocking) in a humoristic christmas song while performing the task. From a technical perspective, the procedure (Figure 4.18) realizes yet another interaction primitive with garments, namely *putting sth. in*. It consists of six basic steps, the first three being identical to those in the coat-check task, the other three implementing the interaction with the objects to be put into the stocking:

[15] `https://youtu.be/2naQbWa-Sv8`

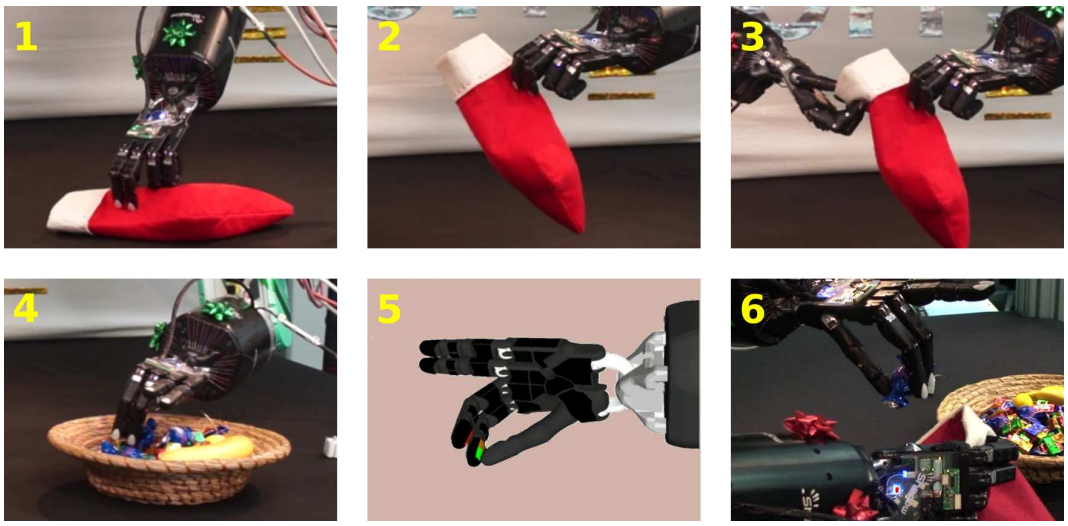



Figure 4.18: The robot christmas elf task (putting candy into a stocking) divided into a sequence of six actions.

1. *Picking up:* The robot picks up the stocking from the tabletop.

2. *Visually observing:* One robot hand holds the stocking in such a way that the boundary can be detected.

3. *Regrasping around the boundary:* Having determined a good boundary grasp pose, the robot regrasps the stocking with the other hand.

4. *Picking up a candy:* The robot hand that was used for the initial grasp releases the stocking and reaches into a gift basket trying to grasp a piece of candy.

5. *Evaluating the grasp:* The grasp is evaluated using the tactile sensors at the fingertips, and the previous step is repeated if the grasp is detected to have failed.

6. *Dropping:* The robot moves its hand above the opening of the stocking and drops the candy into it.

## 4.4 Discussion

In this chapter, we have explored the limits of static representations in robotic perception and action in the clothing domain. By doing so, we have avoided

modeling the complex dynamics of highly deformable objects. However, algorithms that use static models cannot draw on previous states to detect local changes of the environment (such as garment deformations), but instead, they essentially have to search for relevant structures in the whole scene. Therefore, it is important to choose an appropriate input space in which to perform the search. In our case, depth images and point clouds were mostly preferred to color images because we focused on topological and geometric properties rather than the overall appearance of clothes. We have demonstrated that, exploiting prior knowledge of the objects in the scene, the search space can be reduced using a semantic point cloud filter which we have shown to be functionally complete and efficiently parallelizable.
Although we considered static scenes, the representation structure was chosen with an eye toward usefulness for complex interactions and dynamic model initialization. Hence, our focus was on those features of clothes that do not vanish under typical deformations. In particular, we have presented and discussed two methods for boundary component detection. The first method uses polygonal templates of different types of clothing to find all boundary components of a garment that has been spread out flat on a support surface. In an evaluation against the ground truth from a human subject, we found that the performance of the algorithm was acceptable.
However, the static boundary component models of flattened garments turned out to be not so well suited for manual interaction. Therefore, in the second (interactive) method, a robot hand is assumed to pick up and hold the garment such that one of the openings is visible to the camera and the boundary component can be detected using a graph-based algorithm. We have shown that the resulting model can be used for defining a boundary grasp configuration that takes into account grasp comfort, perceptual reliability, and the shape of the boundary component. Our robot applied the boundary grasp in two demonstration procedures implementing the interaction primitives of *hanging up* and *putting sth. in*.

# 5 Reducing the Problem Space for Tracking

Deriving contact relations between a garment and other physical entities (such as a robot hand) from the visible effects is a key element of perception in the clothing domain, as already pointed out in Chapter 2. However, vision algorithms based on static images usually *assume* rather than *detect* that two objects are in contact. For example, if in the interactive perception scheme from Section 4.3 the knit cap slips out between the fingers when the robot tries to lift the garment from the tabletop, that will not be realized until boundary component detection fails. Therefore, to be able to determine at any time whether the garment moves and deforms in a plausible and desired manner when interacting with it (e.g., whether it moves along with the grasping hand), a dynamic model is required.

At the beginning of this chapter, we consider different approaches to modeling the dynamic behavior of non-rigid objects (Section 5.1). In this connection, it will become apparent that tracking highly deformable articles of clothing is extremely difficult, especially if no exact reference model of the garment considered is available. This is due to the vast number of degrees of freedom of clothes in addition to the noisy and incomplete measurements robots have to cope with. Therefore, we suggest reducing the problem space by focusing on the relevant object parts and structures which are, in robot-assisted dressing as well as in many other interaction tasks with clothes, the openings and their spatial relationships. Specifically, in Section 5.2, we build on the models from Chapter 4 and represent openings as closed, oriented chains of movable points which we refer to as *Active Boundary Component Models (ABCMs)*. Compared with the hardly predictable motions of an overall piece of clothing, relatively strict assumptions regarding the dynamics of these models can be made. We express these assumptions through position-based constraints which significantly restrict the degrees of freedom. In Section 5.3, we show how ABCMs as well as *Active Skeleton Models* linking the ABCMs can be initialized, and how they can be tracked visually using point cloud data. Additionally, we consider the task of sliding a rod through a pant leg as a first step toward robotic dressing assistance for physically handicapped persons.

## 5.1 Dynamic Models of Deformable Objects

In the following, we discuss several ways to model the dynamic behavior of a deformable object, i.e., how the position and configuration change over time when the object is manipulated by a human or robot. In general, the configuration of an object can be parameterized in many different ways, but often, it is approximated by a deformable mesh or just a set of points. When we refer to a dynamic model, we do not necessarily mean a model that explicitly includes the acting forces or even the motion parameters (i.e., the velocities and accelerations) of the points. However, what all methods discussed below have in common is that they update a (deterministic or probabilistic) model at every time step depending on some input data. In Section 5.1.1, we consider the case where the input is a sequence of color images, depth images, or point clouds. The approach described in Section 5.1.2 originally assumed that the input is a set of external forces acting upon the model points, but we will use the same constraint solving technique in the context of tracking garment deformations with point cloud data.

### 5.1.1 Models for Visual Tracking

In the past, depth cameras were expensive and most robots were only equipped with a color camera. Therefore, a common simplification of the deformable object tracking problem has been to consider the 2-dimensional projection onto the image plane. But some of the key ideas of the methods below can, at least in principle, be transferred to the 3D case. A simple appearance-based tracking technique is *mean shift* [137] which in most implementations uses a histogram (e.g., a color histogram) as a feature descriptor for the object to be tracked. Since histograms are deformation invariant, they can be applied to non-rigid-objects. However, the mean shift algorithm cannot be used to track object deformations but only positions in the image.

One of the most influential works on tracking deformable curves is the one from Kass et al. [138]. They suggested an energy minimization approach where the energy function is the sum of an image term (e.g., based on edges) and an internal energy term that imposes constraints on the shape of the curve. The models are called *snakes* or *Active Contour Models*, which served as a basis for naming our *Active Boundary Component Models* (Section 5.2). A drawback of snakes is that they do not necessarily show physically plausible behavior, e.g., they tend to shrink over time.

A widely used probabilistic method is the *Condensation (Conditional Density Propagation)* algorithm [139] which is an example of a particle filter based tracker. In the Condensation framework, a stochastic model of the dynamics

of the parameterized object contour has to be learned from training data. In recent years, deep learning has been used to model both object appearances and dynamics for visual tracking [140].
Modeling and tracking the complete configuration of a 3-dimensional object in real time is a very difficult task. Therefore, some methods simplify the problem by relying on visual markers on the object surface. Elbrechter et al. [141] presented an approach to tracking a sheet of paper with fiducial markers printed on it. They employed an external physics engine and a simple P-controller establishing a link between the physical model and the visual input. Schröder et al. [142] suggested a data-driven method for hand tracking with a color glove.
In object tracking with point clouds, a well-established approach that does not require markers or physical simulation is to perform a point matching procedure between the model and the observation points at each frame. *Iterative Closest Point (ICP)* [143] is a comparatively simple point set registration algorithm and will be explained in Section 5.3.2. ICP is mainly suited for rigid object tracking, but extensions for articulated [144] and non-rigid objects [145] exist. It is also possible to define the geometry of the object to be tracked implicitly by its *signed distance function (SDF)*. Then, tracking means finding the object configuration that minimizes the mean square of the SDF over all observation points at the current time step. In the *DART* framework [146], the SDF representation has been extended to articulated objects.
In the hand tracking method described in [147], an energy function consisting of various data fitting and prior terms is minimized using the Levenberg-Marquardt algorithm. Schulman et al. [148] proposed a probabilistic model in which inference is performed by means of an expectation-maximization algorithm where the maximization step involves calls to a physics simulation engine. To the best of our knowledge, the mentioned markerless approaches are either restricted to articulated objects with comparatively few degrees of freedom or tracking has been shown to be robust and accurate only if the object topology is rather simple (such as that of a rope or a piece of cloth).

### 5.1.2 Position-based Dynamics

To achieve physical plausibility, many methods for visual tracking of deformable objects limit the degrees of freedom implicitly through energy terms or by choosing a certain model parameterization. Other approaches rely on complex physical simulation by an external engine which is usually treated as a black box. However, we think that the key element that guarantees physically realistic models is often a constraint solver. Therefore, we now describe a sim-

ple position-based constraint solving technique which has been suggested by Müller et al. in the context of their *Position-based Dynamics (PBD)* framework for computer graphics [149].
In the original PBD paper, the update procedure at each time step is as follows: The model points have masses which, given the current external forces and a set of damping coefficients, are used to generate new velocity and position hypotheses. Then, the positions are corrected such that the object's internal constraints are satisfied. After this constraint projection step, the velocities are changed accordingly. In the following, we ignore the forces and velocities and assume that the points have equal masses.
Position-based constraint projection is an iterative process in which the positions are corrected repeatedly for all unsatisfied constraints, one after another. Let $C$ be a constraint function and $\nabla C$ the gradient with respect to the points affected by the constraint. Furthermore, let $p$ be the concatenation vector of these points. Then, constraint solving means finding a correction $\Delta p$ such that $C(p + \Delta p) = 0$ (equality constraint), $C(p + \Delta p) < 0$, or $C(p + \Delta p) > 0$ (inequality constraints). For equality constraints, this can be approximated by

$$C(p + \Delta p) \approx C(p) + \nabla C(p) \cdot \Delta p = 0. \tag{5.1}$$

The inequality constraint case is not considered separately because the procedure is the same except that it is only performed if the inequality is not satisfied.
Now, $\Delta p$ is restricted to be along $\nabla C(p)$:

$$\Delta p = \lambda \nabla C(p) \tag{5.2}$$

The rationale behind this is twofold: (i) Rigid body modes (translation and rotation) do not change the value of an internal constraint function, i.e., the direction of maximal change $\nabla C(p)$ is perpendicular to rigid body modes. (ii) Corrections $\Delta p$ should be chosen to be independent of rigid body modes.
Equation (5.2) is substituted into Equation (5.1):

$$C(p) + \nabla C(p) \cdot \lambda \nabla C(p) = 0 \tag{5.3}$$

Solving for $\lambda$ yields

$$\lambda = -\frac{C(p)}{|\nabla C(p)|^2}. \tag{5.4}$$

After substituting back into Equation (5.2) and introducing a stiffness coefficient $c_s \in [0, 1]$ specifying the relative strength of the constraint, we have

$$\Delta p = -c_s \frac{C(p)}{|\nabla C(p)|^2} \nabla C(p). \tag{5.5}$$

For the correction of an individual point, that means

$$\Delta p_i = -c_s \frac{C(p)}{\sum_j |\nabla_{p_j} C(p)|^2} \nabla_{p_i} C(p). \tag{5.6}$$

## 5.2 Active Boundary Component Models (ABCMs)

Tracking deformable objects is a hard problem because of the huge amount of degrees of freedom to be modeled. In the case of clothing, the task is complicated further by the complex topology and geometry of most garments. Therefore, we present a reduced approach that is along the lines of Chapter 4 in that only the boundary components are modeled instead of the full garment geometry. We extend this idea to the dynamic case and suggest an integrated representation which takes account of how the openings are topologically linked inside the garment.

### 5.2.1 Definition and Properties

We define an *Active Boundary Component Model (ABCM)* as a tuple of 3D points $p_1, ..., p_N$ with attached constraints. Adjacent points (i.e., $p_i$ and $p_{i+1}$ for $i < N$ as well as $p_N$ and $p_1$) are implicitly considered as being connected by lines forming a piecewise linear approximation of the closed boundary curve. The order in which the points are given specifies the direction of the corresponding garment opening in accordance with the following right-hand rule: If the curled fingers mimic the orientation of the boundary curve, then the interior of the garment is roughly in direction of the thumb (i.e., if one looks inside the opening, the points are in clockwise order).
In ABCMs, external influences on the models (e.g., based on image or point cloud data) and internal priors are kept strictly separate. At each update iteration, the models undergo a two-step process. First, the positions of the boundary component points are manipulated according to the input from the robot's sensors. Then, a position-based constraint solving step ensures that the models fulfill a set of conditions. Thus, ABCMs provide an intuitive and extensible mechanism to express physical, geometric, and topological prior knowledge about clothes in an explicit manner. Specifically, ABCMs have the following properties:

*Simplicity:* Since boundary components are simply approximated by closed sequences of points, no complex curve parameterization is required.

*Flexibility:* ABCMs are in principle not restricted to visual tracking. The model points can be freely manipulated according to whatever forces or sensory modalities influence the object perception.

*Plausibility:* Despite their flexibility, the models behave plausibly in terms of the assumptions made. The assumptions are formalized through constraints on the model points.

*Stability:* Formulating the constraints in a position-based manner ensures high stability and controlability as compared with force-based methods.

One of the basic ideas of ABCMs is that even though the overall dynamics of clothes may be hard to predict, we can still make a few reasonable assumptions regarding the boundary components, in particular the following:

1. *Roughly constant arc length:* Boundary component shrinkage or expansion is minimal during typical garment manipulations.
2. *Smoothness:* Although different materials allow different degrees of deformation, model plausibility is significantly increased by assuming a minimum boundary smoothness.
3. *No entanglement:* The boundary components of many real-world garments (including those considered in our experiments) do not excessively coil out of the plane.

To formalize these assumptions, we derive suitable constraint functions along with their gradients as required by Equation (5.6). We emphasize again that these constraint functions are absolutely independent of any sensory input.

### 5.2.2 Distance Constraints

The first and probably the most important constraints we impose on the boundary component models relate to the distances between adjacent points (Figure 5.1). These constraints mainly have an effect on the local behavior of the boundary curve, but they can also be used to indirectly control global properties. For example, if the number of model points is assumed to remain unchanged and the distances between adjacent points are kept constant, the overall length of the curve is constant, too. In some sense, the smoothness constraint introduced below can also be considered a distance constraint on each three adjacent model points.

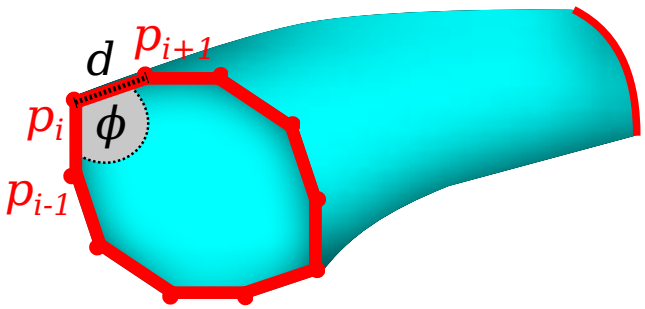



Figure 5.1: Schematic diagram of an ABCM (red) and the distance constraint parameters (black).

**Arc Length** Our first physical assumption regarding the boundary components of clothes has been that they do not excessively expand or shrink. This is ensured geometrically by imposing equality constraints on the distances of adjacent boundary points $p_i$ and $p_{i+1}$ (with $p_{N+1} = p_1$). The corresponding constraint function is

$$C_{dist}(p_i, p_{i+1}) = |p_i - p_{i+1}| - d, \tag{5.7}$$

where $d$ is chosen to be the distance between the two points at initialization time. It is also possible to allow stretching by replacing the equality constraint with two inequality constraints indicating the lower and upper distance tolerances. The gradients of $C_{dist}$ are as in [149]:

$$\nabla_{p_i} C_{dist}(p_i, p_{i+1}) = \frac{p_i - p_{i+1}}{|p_i - p_{i+1}|} \tag{5.8}$$

$$\nabla_{p_{i+1}} C_{dist}(p_i, p_{i+1}) = -\frac{p_i - p_{i+1}}{|p_i - p_{i+1}|} \tag{5.9}$$

Substituting Equations (5.7), (5.8), and (5.9) into Equation (5.6) yields

$$\Delta p_i = -\frac{1}{2} c_s (|p_i - p_{i+1}| - d) \frac{p_i - p_{i+1}}{|p_i - p_{i+1}|} \tag{5.10}$$

and

$$\Delta p_{i+1} = \frac{1}{2} c_s (|p_i - p_{i+1}| - d) \frac{p_i - p_{i+1}}{|p_i - p_{i+1}|}. \tag{5.11}$$

It is easy to see that for $|p_i - p_{i+1}| > d$ the points are moved toward each other, and for $|p_i - p_{i+1}| < d$ they are moved away from each other. From now on, we will only give the constraint functions and their gradients rather than the explicit formulas for $\Delta p_i$.

**Smoothness** Our second assumption has been that the boundary components have a minimum degree of smoothness, i.e., that they do not bend too much locally. A natural way to restrict bending is through angular inequality constraints ($C_{smooth}(p_{i-1}, p_i, p_{i+1}) > 0$) between adjacent segments $\overline{p_{i-1}p_i}$ and $\overline{p_i p_{i+1}}$ of a boundary component model (with $p_0 = p_N$ and $p_{N+1} = p_1$), the constraint function being

$$C_{smooth}(p_{i-1}, p_i, p_{i+1}) = \arccos\Big(\big(\underbrace{\tfrac{p_{i-1}-p_i}{|p_{i-1}-p_i|}}_{\hat{a}}\big)^T\big(\underbrace{\tfrac{p_{i+1}-p_i}{|p_{i+1}-p_i|}}_{\hat{b}}\big)\Big) - \phi, \tag{5.12}$$

and the gradients with respect to the points being

$$\nabla_{p_{i-1}} C_{smooth}(p_{i-1}, p_i, p_{i+1}) = -\tfrac{1}{\sqrt{1-(\hat{a}^T\hat{b})^2}}\big((J_{p_{i-1}}\hat{a})^T\hat{b}\big), \tag{5.13}$$

$$\nabla_{p_i} C_{smooth}(p_{i-1}, p_i, p_{i+1}) = -\tfrac{1}{\sqrt{1-(\hat{a}^T\hat{b})^2}}\big((J_{p_i}\hat{a})^T\hat{b} + (J_{p_i}\hat{b})^T\hat{a}\big), \tag{5.14}$$

and

$$\nabla_{p_{i+1}} C_{smooth}(p_{i-1}, p_i, p_{i+1}) = -\tfrac{1}{\sqrt{1-(\hat{a}^T\hat{b})^2}}\big((J_{p_{i+1}}\hat{b})^T\hat{a}\big). \tag{5.15}$$

To compute the gradients, the following Jacobians are required:

$$J_{p_{i-1}}\hat{a} = \frac{I_3 - \left(\frac{p_{i-1}-p_i}{|p_{i-1}-p_i|}\right)\left(\frac{p_{i-1}-p_i}{|p_{i-1}-p_i|}\right)^T}{|p_{i-1}-p_i|} \tag{5.16}$$

$$J_{p_i}\hat{a} = \frac{-I_3 + \left(\frac{p_{i-1}-p_i}{|p_{i-1}-p_i|}\right)\left(\frac{p_{i-1}-p_i}{|p_{i-1}-p_i|}\right)^T}{|p_{i-1}-p_i|} \tag{5.17}$$

$$J_{p_{i+1}}\hat{b} = \frac{I_3 - \left(\frac{p_{i+1}-p_i}{|p_{i+1}-p_i|}\right)\left(\frac{p_{i+1}-p_i}{|p_{i+1}-p_i|}\right)^T}{|p_{i+1}-p_i|} \tag{5.18}$$

$$J_{p_i}\hat{b} = \frac{-I_3 + \left(\frac{p_{i+1}-p_i}{|p_{i+1}-p_i|}\right)\left(\frac{p_{i+1}-p_i}{|p_{i+1}-p_i|}\right)^T}{|p_{i+1}-p_i|} \tag{5.19}$$

The actual smoothness of the model is influenced not only by $\phi$, but also by the number of points $N$. Therefore, we make the choice of $\phi$ in Equation (5.12) dependent on an overall smoothness parameter $k_{smooth} \in [0, 1]$. The smoothest possible boundary component model ($k_{smooth} = 1$) is a regular $N$-polygon in which all internal angles have the same value $\phi = \frac{(N-2)}{N} \cdot \pi$. Hence, we set

$$\phi = \frac{(N-2)}{N} \cdot \pi \cdot k_{smooth}. \tag{5.20}$$

Figure 5.2(a) shows an ABCM that violates the smoothness assumption. In Figure 5.2(b), the same model is shown after smoothness constraint projection (with $k_{smooth} = 0.8$).

### 5.2.3 Entanglement Constraints

While constant length and smoothness have been ensured simply by imposing distance constraints on adjacent model points, restricting entanglement is less straightforward. One possibility could be to constrain the local torsion of the boundary curve. For example, Umetani et al. [150] have shown how, within the PBD framework, all three components of bending and twisting of an elastic rod can be controlled independently using so-called ghost points that represent the rod's material. As an alternative, we discuss how coiling out of the plane can be constrained globally and without giving up the idea of one-dimensional boundary component models. To this end, we define two different constraint functions which act upon all model points simultaneously.

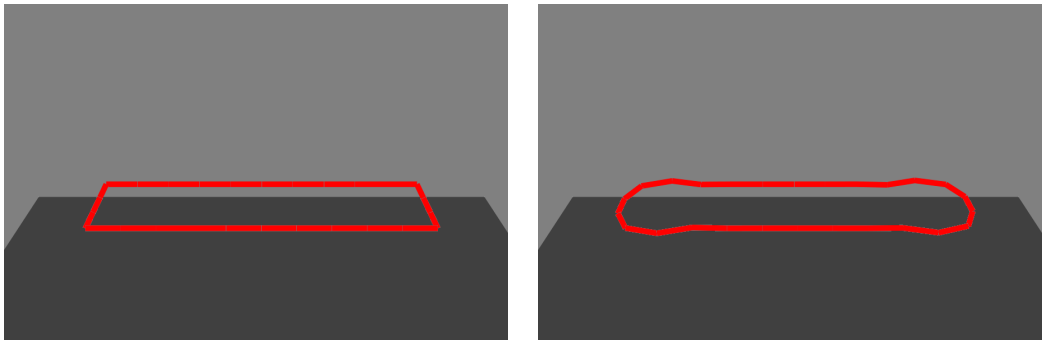
**Writhe** One way to quantify the entanglement of a boundary component model is through the *writhe* which can be viewed as the sum of all signed self-crossings averaged over all possible viewing directions. The writhe of a simple, closed, differentiable curve $\gamma$ with points $r_1$ and $r_2$ along the curve is defined as the Gauss integral

$$Wr = \frac{1}{4\pi} \int_\gamma \int_\gamma dr_1 \times dr_2 \cdot \frac{r_1 - r_2}{|r_1 - r_2|^3}. \tag{5.21}$$

A similar definition could be used to calculate the writhe of a piecewise linear curve. However, it is computationally more efficient to consider an additional virtual curve $\gamma'$, to calculate both the twist $Tw$ and the Gauss linking number $Lk$ of $\gamma$ with $\gamma'$, and to employ the Călugăreanu-White-Fuller theorem [151]:

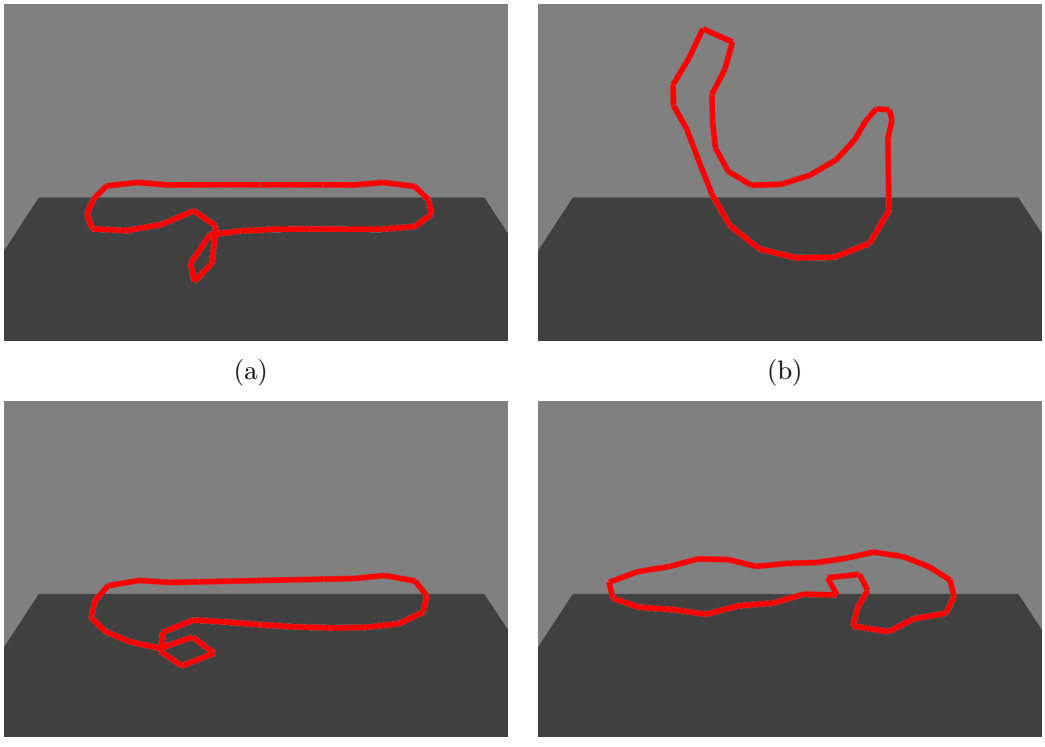
$$Wr = Lk - Tw \tag{5.22}$$

The linking number $Lk$ is a topological invariant and an integer indicating how often two closed curves wind around each other. The twist $Tw$ is neither an integer nor a topological invariant, but it measures the rate of rotation of $\gamma'$ around $\gamma$. Thus, Equation (5.22) describes geometrically how a ribbon with boundary curves $\gamma$ and $\gamma'$ reduces torsional stress by forming coils, i.e., by converting twist into writhe. For details and a comparison of several methods to calculate the writhe, the reader is referred to [152]. The gradient of the writhe can be computed using the formula from [153].
Now, we are able to define a writhe constraint function:



(a) (b)

Figure 5.2: Smoothness constraint experiment. (a) Initial rectangular ABCM. (b) The ABCM after constraint projection.



(a) (b)

(c) (d)

Figure 5.3: Entanglement constraint experiment. (a) Initial coiled ABCM. (b) The ABCM after writhe constraint projection. (c) The ABCM after planarity constraint projection. (d) The ABCM after writhe and planarity constraint projection.

$$C_{writhe}(p) = |Wr(p)| - d_{writhe}, \tag{5.23}$$

where $d_{writhe}$ denotes the maximum entanglement in an inequality constraint ($C_{writhe}(p) < 0$).
The writhe quantity has a few disadvantages. First, it is not defined for self-intersecting boundary component models, which can be circumvented by imposing some additional distance constraints on non-adjacent points. Second, under certain circumstances, the gradient-based solver may not converge to a root of the writhe constraint function. Third, not only the writhe of any planar curve but also that of any curve located on a sphere is zero. Therefore, ABCMs tend to form highly non-planar configurations during uncoiling. To illustrate this, we initialized a coiled ABCM with $Wr = 0.74$ (Figure 5.3(a)). After writhe constraint projection with $d_{writhe} = 0.5$, $Wr$ is indeed reduced to 0.5, but the global configuration of the model has changed drastically (Figure 5.3(b)).

**Planarity** To avoid the above-mentioned issues with the writhe constraint function, coiling out of the plane can also be restricted by imposing a planarity constraint on the model. The cross product version of the so-called *shoelace formula* yields a vector $n_p$ which is perpendicular to and whose length is twice the area of a given polygon. For non-planar polygonal chains such as *ABCMs*, the result is an approximate "best-fit" normal vector:

$$n_p = \sum_{i=1}^{N} p_i \times p_{i+1} \tag{5.24}$$

with $p_{N+1} = p_1$. Let $\hat{n}_p = \frac{n_p}{|n_p|}$ be the unit normal, and let $m_p$ be the mean of the model points. Then, a planarity constraint function can be defined as

$$C_{plan}(p) = \frac{1}{N} \sum_{i=1}^{N} \big( \underbrace{\hat{n}_p^T (p_i - m_p)}_{d_i} \big)^2 - d_{plan}, \tag{5.25}$$

limiting the mean squared distance of the points to a plane when used in an inequality constraint ($C_{plan}(p) < 0$). The gradients with respect to the points are given by

$$\nabla_{p_j} C_{plan}(p) = \frac{2}{N} \left[ \sum_{i=1}^{N} d_i (J_{p_j} \hat{n}_p)^T (p_i - m_p) - \sum_{i=1, i \neq j}^{N} \tfrac{1}{N} d_i \hat{n}_p + (1 - \tfrac{1}{N}) d_j \hat{n}_p \right] \tag{5.26}$$

with

$$J_{p_j}\hat{n}_p = \frac{1}{|n_p|}J_{p_j}n_p - \hat{n}_p\hat{n}_p^T J_{p_j}n_p \tag{5.27}$$

and

$$J_{p_j}n_p = \begin{pmatrix} 0 & p_{j+1_z} - p_{j-1_z} & p_{j-1_y} - p_{j+1_y} \\ p_{j-1_z} - p_{j+1_z} & 0 & p_{j+1_x} - p_{j-1_x} \\ p_{j+1_y} - p_{j-1_y} & p_{j-1_x} - p_{j+1_x} & 0 \end{pmatrix}. \tag{5.28}$$

Again, we make $d_{plan}$ in Equation (5.25) dependent on an overall planarity parameter $k_{plan} \in [0, 1]$. From some geometric considerations, we derive that the mean distance of the model points to the fitted plane cannot exceed $\frac{1}{16}$ of the boundary component model's arc length $L$. Hence, we set

$$d_{plan} = \left(\frac{L}{16}(1 - k_{plan})\right)^2. \tag{5.29}$$

The planarity constraint indeed helps to avoid coiling of the boundary component model, but it sometimes also prevents uncoiling when the curve is already in an entangled state. For example, the writhe of the ABCM in Figure 5.3(c) has even increased from 0.74 to 0.98 after planarity constraint projection. Therefore, in practice, it makes sense to have both a writhe constraint and a planarity constraint, and to decrease the stiffness of the planarity constraint during uncoiling. In Figure 5.3(d), it can be seen that applying a writhe constraint with high stiffness ($c_s = 0.75$) and a planarity constraint with significantly lower stiffness ($c_s = 0.05$) results in a flattened and disentangled model with $Wr = 0.03$.

### 5.2.4 Active Skeleton Models

ABCMs as defined so far model the internal degrees of freedom of individual garment openings, but they do not incorporate the relationships between different boundary components. *Active Skeleton Models* extend the constraint-based concept to include a coarse representation of the overall geometry and topology of an article of clothing. We expect that the skeleton models could robustify tracking of the boundary components by providing some information about the relative poses of the openings. Furthermore, they can be used to define trajectories for the human limbs in dressing assistance.
We model the skeletons as star-shaped structures with a single central point, one end point for each opening, and several points describing the paths from the central point to the boundary components. In Section 5.3.1, we will describe

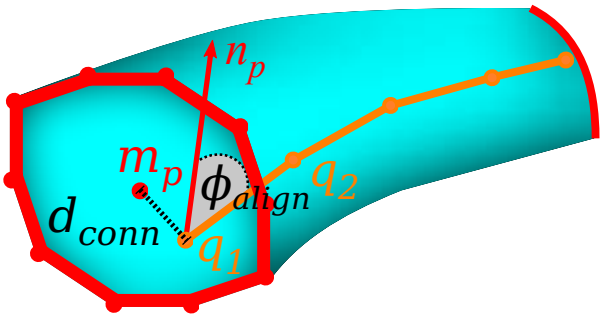



Figure 5.4: Schematic diagram of an Active Skeleton Model (orange) linking two ABCMs (red), and the skeleton constraint parameters (black).

how a skeleton model can be initialized together with the ABCMs. Possible deformations are again formalized by means of position-based constraints. We use internal angular and distance constraints to define the degrees of allowed stretching and bending. The attachments of the ABCMs to the skeleton (Figure 5.4) are characterized by the following two properties:

*Connectedness:* The main purpose of a skeleton model is to link the openings of a garment. Therefore, all skeleton end points are positioned close to the centers of their respective boundary components.

*Alignment:* The openings define the optimal entry directions into the garment interior. Hence, each skeleton end is roughly aligned with the normal vector of the boundary component model it is attached to.

**Connectedness** We achieve connectedness between the skeleton model and an ABCM by imposing a distance constraint on the skeleton end point $q_1$ and the boundary points $p$. Assuming that the mean $m_p$ is a sufficient approximation of the boundary component center, we can define a simple constraint function

$$C_{conn}(p, q_1) = |m_p - q_1| - d_{conn} \tag{5.30}$$

with gradients

$$\nabla_{q_1} C_{conn}(p, q_1) = -\frac{m_p - q_1}{|m_p - q_1|} \tag{5.31}$$

and

$$\nabla_{p_i} C_{conn}(p, q_1) = \frac{m_p - q_1}{N|m_p - q_1|}, \tag{5.32}$$

where $d_{conn}$ is usually set to 0 in an equality constraint.

**Alignment** Each end of the skeleton model is aligned with the normal vector of the corresponding ABCM by means of an angular constraint on the last skeleton segment $\overline{q_1 q_2}$ and the unit normal $\hat{n}_p$ of the boundary component:

$$C_{align}(p, q_1, q_2) = \arccos\left((\hat{n}_p)^T (\underbrace{\tfrac{q_2 - q_1}{|q_2 - q_1|}}_{\hat{e}})\right) - \phi_{align} \tag{5.33}$$

The gradients with respect to the boundary points are

$$\nabla_{p_i} C_{align}(p, q_1, q_2) = -\tfrac{1}{\sqrt{1-(\hat{n}_p^T \hat{e})^2}} \left((J_{p_i} \hat{n}_p)^T \hat{e}\right), \tag{5.34}$$

where $J_{p_i} \hat{n}_p$ is the Jacobian of the unit normal from Equation (5.27).
The gradients with respect to the skeleton points are given by

$$\nabla_{q_j} C_{align}(p, q_1, q_2) = -\tfrac{1}{\sqrt{1-(\hat{n}_p^T \hat{e})^2}} \left((J_{q_j} \hat{e})^T \hat{n}_p\right), \tag{5.35}$$

for $j \in \{1, 2\}$.
The formulas for the required Jacobians are as follows:

$$J_{q_1} \hat{e} = \frac{-I_3 + \left(\frac{q_2 - q_1}{|q_2 - q_1|}\right)\left(\frac{q_2 - q_1}{|q_2 - q_1|}\right)^T}{|q_2 - q_1|} \tag{5.36}$$

$$J_{q_2} \hat{e} = \frac{I_3 - \left(\frac{q_2 - q_1}{|q_2 - q_1|}\right)\left(\frac{q_2 - q_1}{|q_2 - q_1|}\right)^T}{|q_2 - q_1|} \tag{5.37}$$

The parameter $\phi_{align}$ is used in an inequality constraint ($C_{align}(p, q_1, q_2) < 0$) to specify the misalignment tolerance.

## 5.3 Tracking ABCMs with Point Clouds

Under the assumption that the position-based constraint solving step ensures that the physical, topological, and geometric conditions are satisfied, the model points can be moved according to input from the robot's sensors in a rather unrestricted way. We think that, in principle, various types of sensory input could be used. However, in the present work, we focus on point cloud data from a Kinect sensor. In the following, we show how ABCMs and skeleton

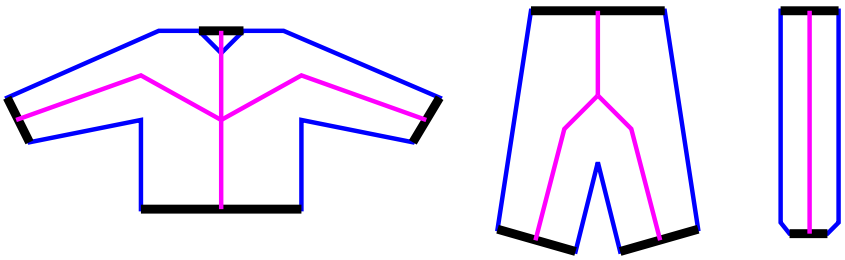


Figure 5.5: Template polygons and skeletons (pink) of a sweater, a pair of pants, and a legwarmer.

models can be initialized, and how they can be tracked by means of an iterative edge point matching approach. The method has been evaluated with different articles of clothing and applied to a simplified robotic dressing assistance task.

### 5.3.1 Initialization

In principle, both the polygon-based boundary component detector from Section 4.2 and the interactive perception approach from Section 4.3 can be used to initialize ABCMs. The interactive method is mainly suited for garments with only one opening, i.e., for initializing a single ABCM. The only adjustment we make to the algorithm is to distribute the model points evenly along the detected boundary curve (at a distance of 1 cm) rather than to employ a curvature-based corner detector. Similarly, in the polygon-based method, the points are equidistantly placed on the resulting rectangular boundary component model. The violation of the smoothness assumption is going to vanish within the first iterations of constraint solving.
In the case of garments with multiple boundary components, a representation of how the openings are linked in the interior of the garment is often required. Therefore, we have extended the method from Section 4.2 to be capable of initializing an Active Skeleton Model along with the ABCMs. For this purpose, template skeletons (depicted as pink lines in Figure 5.5) have been added to the polygonal garment prototypes. Our algorithm deforms a copy of the given template skeleton so as to match the geometry of the garment polygon that has been extracted from point cloud data. The central point of the skeleton remains unchanged because we assume it to be fixed with respect to the polygon centroid. Then, the individual skeleton branches are rotated and stretched in such a way that the end points match the centers of the (black) segments representing the 2D projections of the garment openings. The result of this process is visualized for a pair of pants in Figure 5.6. Figure 5.7 shows the skeleton model after its projection onto the tabletop.

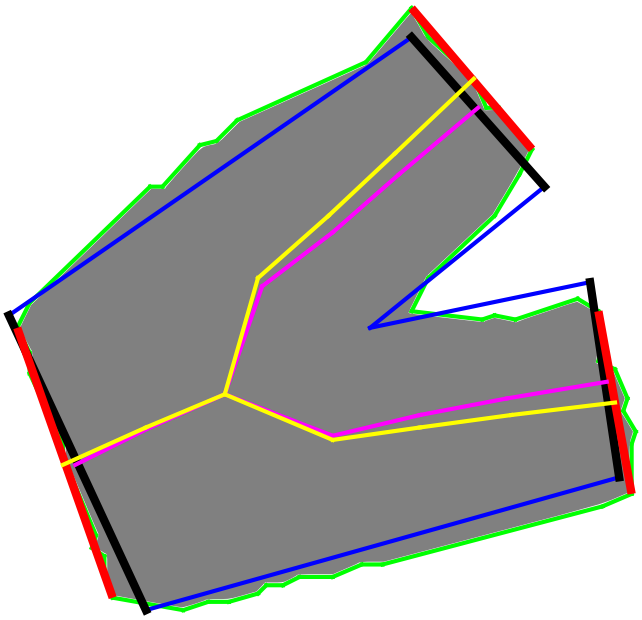


Figure 5.6: Result of the template skeleton deformation. The polygon segments representing the garment openings are depicted as red lines, the corresponding template segments as black lines. The template skeleton and its deformed copy are shown in pink and yellow, respectively.

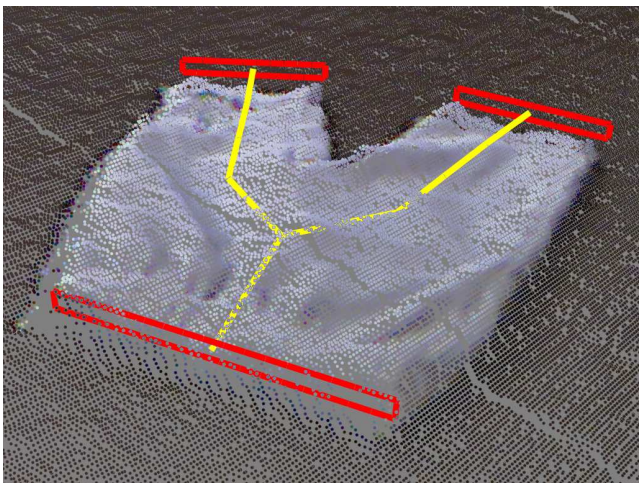


Figure 5.7: Active Skeleton Model (yellow) and three ABCMs (red) right after initialization.

### 5.3.2 Iterative Edge Point Matching

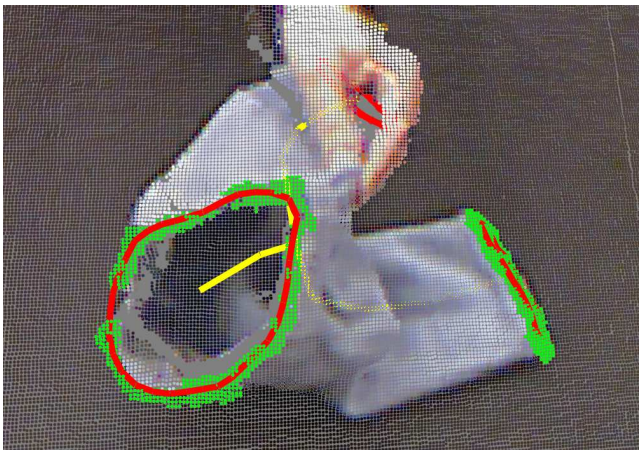
Much like in the graph-based detection approach from Section 4.3, we assume that the boundary components of clothes can be found based on surface edges that, in the ideal case, form closed curves. However, while the static detector has to perform a global search, our tracking algorithm always uses the previous state of the model which is updated only locally. The method can therefore make weaker assumptions (e.g., the garment openings can be less wide open) and estimate the boundary component configurations in real time. At each frame, using edge points extracted from a point cloud, the algorithm first determines how the positions and orientations of the ABCMs have changed. Then, the edge points are used again for a rough heuristic estimate of the model deformations. It is not critical if this estimate is somewhat inaccurate, considering the fact that physical plausibility is subsequently ensured by a set of position-based constraints. In summary, there are three steps that precede constraint projection during ABCM tracking with point clouds:

1. Edge point detection
2. Rigid transformation
3. Heuristic deformation

**Edge Point Detection** Surface edges of sufficiently thick clothes can be detected by employing the technique from Ückermann et al. [132] which finds differences in angle between adjacent surface normals in the point cloud. For details, the reader is referred to Section 4.3.2. We apply the method twice, using slightly different parameters in the vicinity of the tabletop to detect the weak edges of flattened garments. The result is shown in Figure 5.8(a) using the examples of a spread-out and a lifted pair of pants. At each time step, our algorithm collects a set $E$ of relevant edge points by searching in a range around the ABCM points. We define two search radii, a smaller one in direction of the boundary component's normal vector (pointing inside the opening), and a larger one in the opposite direction. This accounts for the fact that the boundary components are located at the outer ends of a garment. The extracted edge points are depicted in green in Figure 5.8(b).

**Rigid Transformation** The internal constraints of an ABCM serve as prior knowledge in estimating its deformations, but translation and rotation have to be detected solely based on sensory data. Specifically, the model points should be roughly aligned with the edge points in $E$. We determine a rigid

(a)



(b)

Figure 5.8: Edge point detection during point cloud based ABCM tracking. (a) Surface normal based edge detection result. (b) Relevant edge points (green) lying within a certain range around an ABCM (red).

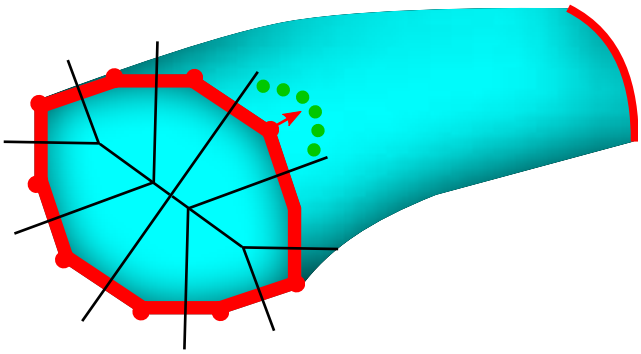


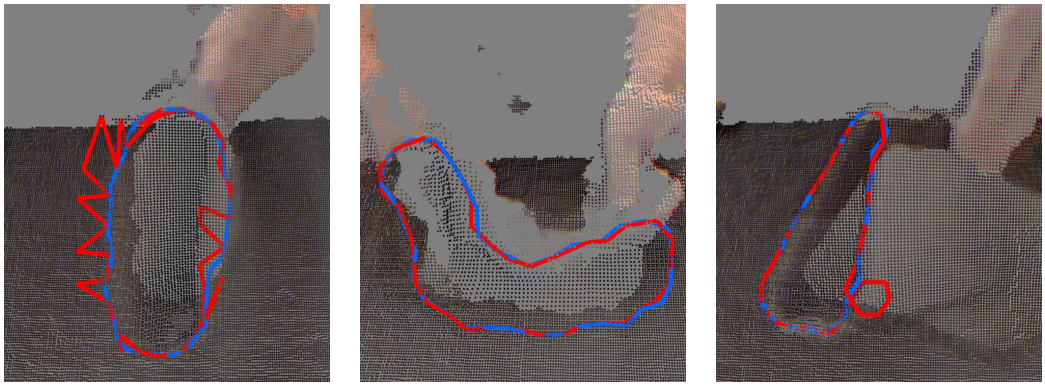
Figure 5.9: Deformation heuristic during point cloud based ABCM tracking. ABCMs are deformed by moving the model points (red) toward the means of the edge points (green) in their respective Voronoi regions. The boundaries between the Voronoi regions are indicated by black lines.

transformation of the ABCM using a reverse *Iterative Closest Point (ICP)* approach. The ICP algorithm [143] in its simplest form consists of three basic steps which are performed iteratively until the alignment between the model and the data points does not improve anymore: (i) Assign each model point to its closest data point. (ii) Estimate a rigid transformation which, using the assignment from the previous step, minimizes the mean squared point to point distance. (iii) Transform the model accordingly. Since there are usually many more data points than model points, it is however significantly more effective to find a transformation $T$ from the points in $E$ to the model (i.e., to assign the data points to the model points) and to apply $T^{-1}$ to the ABCM model.

**Heuristic Deformation** Finally, the ABCM is deformed to match $E$ even better. To this end, another closest point assignment step between the data points and the model points is performed. This is equivalent to decomposing the 3D space into *Voronoi* regions using the rigidly transformed model points before shifting the model points toward the centroids of the edge points that fall within their respective regions (Figure 5.9). To increase model stability, we limit the displacement of an individual point per time step to the distance between two adjacent model points.
We emphasize that this simple heuristic approach to estimating deformation neither considers any of our model assumptions nor does it update the skeleton estimates. Therefore, boundary component tracking relies on effective constraint solving. We note that, in our current implementation, there is no partial occlusion handling, i.e., tracking of an individual ABCM stops immediately when another object crosses the view ray of a model point.

### 5.3.3 Evaluation

Tracking with point clouds was again implemented using ICL and runs in real time (30 Hz) on a PC with a modern graphics card. We used a knit cap, i.e., a garment with a single boundary component, to qualitatively evaluate the influence of individual constraints on the tracking performance. In each trial, we initialized two ABCMs with different parameters using the graph-based method from Section 4.3, and observed how the models behaved during manipulation by a human. First, we tested an unconstrained model against an ABCM without stretching tolerance, $k_{smooth} = 0.5$, $k_{plan} = 0.85$, and a maximum writhe of 0.25. It can be clearly seen in Figure 5.10(a) that the unconstrained model heavily violates our assumptions, while the constrained model reflects the real boundary component fairly well. In a second trial (Figure 5.10(b)), we compared the constrained ABCM from the first trial with



(a) (b) (c)

Figure 5.10: Comparison of the ABCM tracking performance between several model configurations. (a) Unconstrained (red) vs. constrained (blue). (b) Without (red) vs. with entanglement constraints (blue). No stretching allowed. (c) Without (red) vs. with entanglement constraints (blue). Stretching allowed.

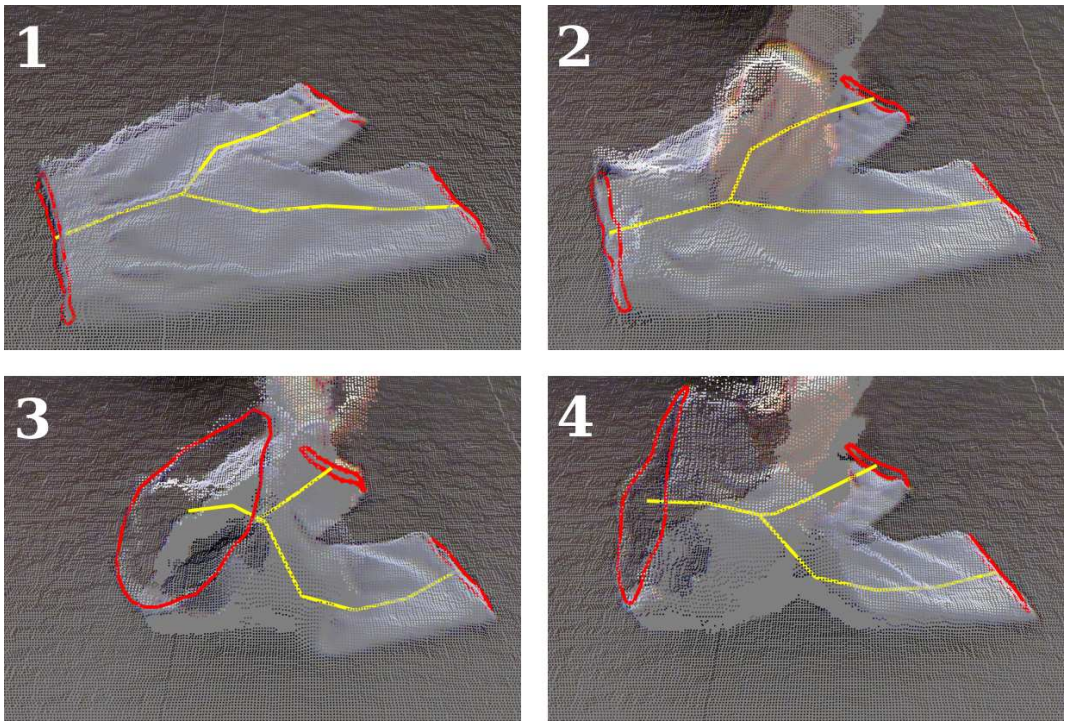



Figure 5.11: Point cloud based ABCM (red) and skeleton (yellow) tracking of a pair of pants shortly after initialization (1), while grasped (2), lifted (3), and moved (4) by a human hand.

an ABCM from which we removed the planarity and writhe constraints. We found that the performance of both models was similar and coiling occurred rarely, even in the condition without entanglement constraints. However, when we allowed stretching (up to 50 percent) in both conditions (Figure 5.10(c)), we frequently observed boundary component entanglement if it was not explicitly constrained.
We also tested the dynamic behavior of Active Skeleton Models with attached ABCMs using the articles of clothing from the test set presented in Section 4.2. Figure 5.11 shows tracking of a pair of pants. We found that the models reacted plausibly to several manipulations such as grasping, lifting, moving, or slightly deforming parts of the garments. The models were indeed not robust against strong occlusions or deformations such as folding a sleeve, but we emphasize that there was no visual tracking of the skeletons or the overall garments. Taking into account that the skeleton models only followed the boundary component dynamics in a constraint-based manner, they represented the object configurations surprisingly well.

### 5.3.4 Application to a Simplified Dressing Task

We also investigated the performance of our model in a controlled robotic scenario that contained several elements of the dressing assistance task. As already pointed out in Section 2.3, the task of getting dressed consists of three basic action patterns: generating a suitable initial configuration, sliding the limbs through the garment interior, and pulling the garment over the limbs. In the current experiment, we focused on the first two patterns. Specifically, the robot's task was to slide a rod through a pant leg, which can be regarded as an abstraction of dressing a leg prosthesis. The task consisted of two steps (Figure 5.12):

*Increasing the opening size:* We specified a heuristic grasp position (the highest point in a region behind the opening) which allowed the robot to slightly lift the garment from the tabletop in order to increase the size of the area circumscribed by the boundary component $b_0$ (i.e., the waist opening through which the rod has to pass). Success was measured using the *relative opening size* $S \in [0, 1]$ which we defined as the ratio of the area to its upper bound (approximated by the area $\frac{L^2}{4\pi}$ of a circle):

$$S = \frac{2\pi|n_p|}{L^2}, \tag{5.38}$$

where $L$ is the arc length and $n_p$ is the normal vector from Equation (5.24) whose length is twice the area of the ABCM.

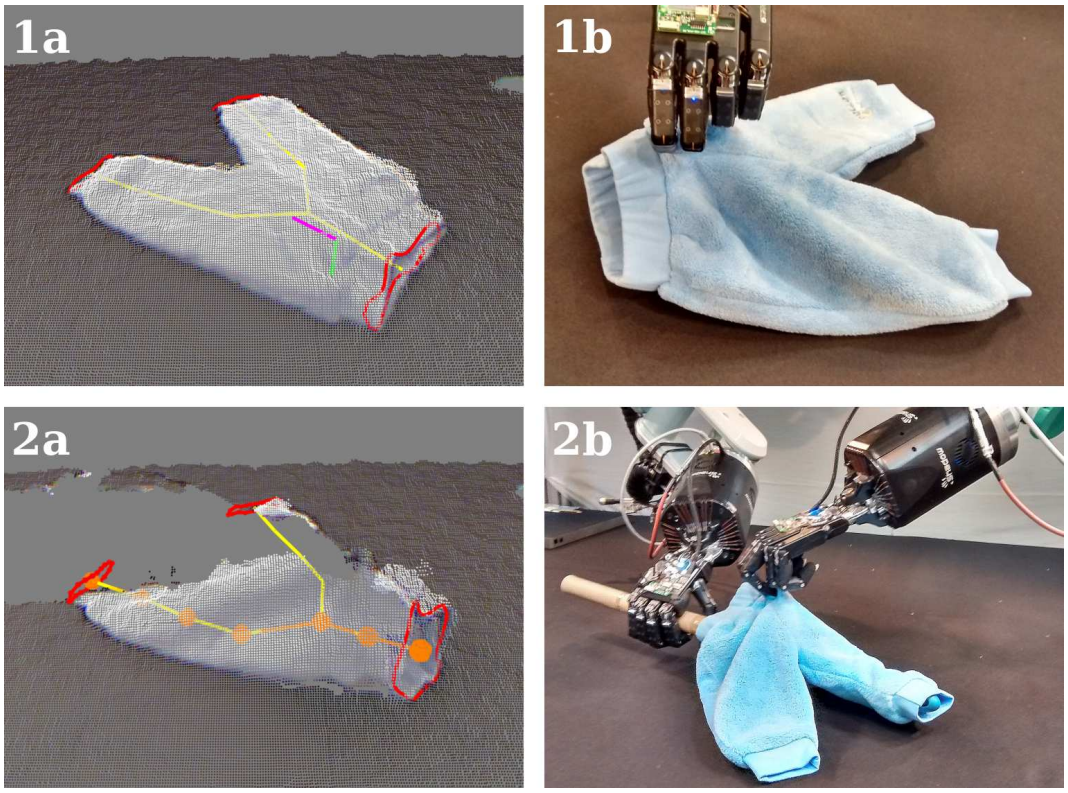



Figure 5.12: Bimanual robot sliding a rod through a pant leg. (1a) Heuristic grasp pose detection. The approach vector is depicted in green, the orientation is shown in pink. (1b) The left hand lifting the garment in order to increase the size of the opening. (2a) Trajectory detection (orange) based on the skeleton model of the garment. (2b) The right hand pushing the rod through the garment interior. The tip of the rod follows the detected trajectory.

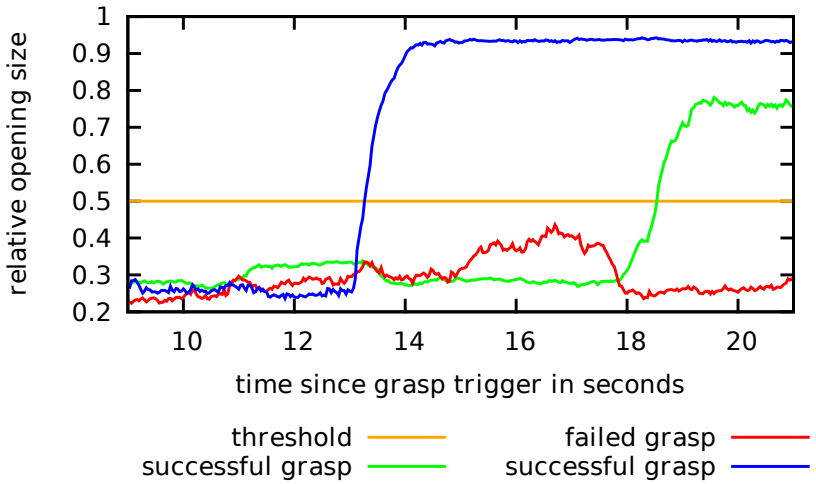



Figure 5.13: Time curves of the relative opening size $S$ of boundary component $b_0$ (waist opening) during the simplified dressing assistance experiment.

*Following the trajectory:* If the waist opening $b_0$ was wide enough ($S > 0.5$), the robot used the other hand to slide the rod through the boundary component and the interior of the garment toward the target leg opening. As suggested in Section 2.3, the rod was integrated into the robot's kinematic chain, and its tip followed a static path along the skeleton model while the orientation of the rod was aligned with the last segment of the trajectory.

The robot performed the task successfully with both pants from our test set. However, in the second run, the initial grasp was not strong enough, and the garment slipped out of the robot's hand. Our system correctly detected the failure by checking the relative opening size $S$, triggered a new grasp, and completed the task. The time curves of $S$ for the failed grasp and both successful grasps are shown in Figure 5.13. Despite the lack of force control, the tip of the rod did not get stuck in the fabric. This indicates that the skeleton models provided suitable paths through the garment.

## 5.4 Discussion

In this chapter, we have suggested a novel method for modeling how the configuration of a garment changes over time during human or robotic manipulation. While capturing the full dynamics of such complex objects as articles

of clothing remains an almost intractable problem, we have presented a reduced approach using deformable models of the boundary components which we refer to as ABCMs. We were able to show that ABCMs limit the degrees of freedom to a tractable level by imposing position-based constraints on the model points. In our framework, the topological and geometric relationships between the ABCMs are described by so-called Active Skeleton Models. We explained how these skeleton models with attached ABCMs can be initialized using slightly modified versions of the detection schemes from Chapter 4, and how they can be tracked with point cloud data.
In experiments with a bimanual robot, we demonstrated the applicability of the proposed representation to the interaction primitive of *sliding sth. through* a garment, which plays an important part in robot-assisted dressing. In particular, our robot accomplished the task of sliding a rod through a pant leg. An important requirement for performing this task is to increase the size of the area circumscribed by the boundary component at the waist end of the pants such that the rod fits through the opening. This can be achieved by slightly lifting the top part of the pants' fabric. Using an ABCM to estimate the current size of the considered opening, the robot knows at any time whether its grasp is successful and has the desired effect, i.e., it derives the contact relation between the fingers and the garment indirectly from the state of the model. Furthermore, we showed that the configuration of the skeleton model after grasping and lifting the garment can be used to define successful trajectories for the tip of the rod through the garment interior.
In the general case, robotic assistance with dressing obviously demands a number of additional skills, both in perception and manipulation. These skills presumably require the integration of different modalities such as color vision, proprioception, force/torque, or tactile sensing. We are optimistic that, in future work, the independence of our model assumptions from any particular type of sensory input may prove beneficial in this regard.

# 6 Reducing the Problem Space for Policy Search

One reason why humans are so good at handling such complex everyday objects as clothes is that they are experts both in understanding and executing given plans, and in applying more implicitly learned strategies (commonly referred to as policies in reinforcement learning). To be able to carry out a predefined plan, a robot requires an explicit and more or less accurate representation of (the task-relevant part of) the environment. In the previous two chapters, using topology-based models and domain knowledge related to these models, we were able to implement some interactions with clothes, e.g., sliding a thin rod through the interior of a garment. However, another interaction primitive which is essential to the dressing task, namely pulling a garment over a human body part (or any other rigid or non-rigid object), is much more difficult to model explicitly. This is because the physical entities involved (the robot hands, the garment, and the body part to be dressed) are in tight contact most of the time which complicates modeling in two different ways. First, there is usually heavy occlusion, i.e., the most relevant areas of the scene (the areas of contact between the objects) are often invisible to the camera. Second, the effects of robotic motion cannot be estimated so easily. To simulate how a garment deforms and wraps around another object during contact-rich interaction, many parameters such as the friction coefficients and other material properties would be required which are generally not known in advance. Therefore, we believe that, in such cases, it is more effective to learn successful policies from experience gained during interaction with the real environment.

In this chapter, after introducing and discussing some of the key challenges and methods of reinforcement learning and policy optimization (Section 6.1), we will focus on the robotic dressing assistance problem again. In particular, as a step toward solutions for the general task, we consider the example of a bimanual robot that learns to put a knit cap on a styrofoam head (Section 6.2). Our approach avoids modeling the details of the garment and its deformations. Instead, we suggest learning in a reduced head-centric policy space. In Section 6.3, we demonstrate how this low-dimensional policy parameteri-

zation, combined with a suitable objective function for determining the right amount of contact between the knit cap and the head, enables a direct policy search algorithm to find successful trajectories for this task.

## 6.1 Reinforcement Learning and Optimization

While we have already equipped our robot with the ability to evaluate some of its actions from the visible effects, the core problem considered in this chapter is different and can be formulated as follows: Learn successful actions not from what can be seen immediately, but from the long term consequences (in our example task: after releasing the knit cap, it slips off or holds firmly on the head). These consequences can be regarded as performance feedback the environment (or an experimenter) provides to the robot. In this sense, the robot is confronted with a typical *reinforcement learning* problem: It wants to optimize an objective function (representing the long term reward) which is not known in advance but has to be learned by interacting with the environment. Reinforcement learning differs from *supervised learning* in that it does not rely on prespecified examples of correct or incorrect behavior, and it differs from *unsupervised learning* in that the goal is not to find hidden structure in the input data but to maximize future rewards [154].
In the following, we discuss two approaches to solving reinforcement learning problems. First, we give a definition of the most commonly used framework, namely *Markov decision processes (MDPs)*, and we describe how learning takes place in this framework (Section 6.1.1). Then, we consider a quite different class of methods often referred to as *evolution strategies* (Section 6.1.2). It will become apparent that there are similar challenges, such as *credit assignment*, *sample efficiency*, and the *exploration-exploitation dilemma*, in both classes of approaches. We think that looking closely at how different methods tackle these issues helps in understanding why a particular technique is suited to a given problem (e.g., why we have chosen a variant of the CMA-ES algorithm for our example task).

### 6.1.1 Policy Search in MDPs

An MDP is a tuple $(S, A, P, R)$, where $S$ is a set of states, $A$ is a set of actions, $P(s'|s, a)$ is the transition probability that action $a$ in state $s$ will lead to state $s'$, and $R(s, a)$ is the immediate reward for taking action $a$ in state $s$. The problem to be solved in an MDP is to find a *policy* $\pi$ that maximizes the expected return. In episodic tasks with a specified terminal state, the return is usually the sum of the rewards gained during one episode.

In the continuing case, the return $G_t$ from time $t$ until infinity can be defined as $G_t = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$, with the discount factor $\gamma \in [0, 1]$ determining the relative importance of future rewards. A deterministic policy specifies for each state an action to be taken, whereas a stochastic policy assigns to each state and action a probability $\pi(a|s)$ of choosing action $a$ in state $s$. If all parameters of an MDP are given, it can be solved using *dynamic programming*. However, if the transition probabilities or the rewards are unknown, an optimal policy has to be learned [154].

**Value-based Learning** Learning in MDPs can be done using *action-value functions* $q_\pi(s, a)$ that represent the expected return when starting in $s$, taking action $a$, and then following $\pi$. The general idea is to find an optimal policy by alternately changing an approximate value function to better match the true value function of the current policy, and making the policy *greedy* with respect to the current value function (such that in each state, the action with the highest value is selected). In *Monte Carlo methods*, the value function is approximated by sampling and averaging returns after each episode. By contrast, *temporal difference methods*, such as *Q-learning* [155] and *SARSA* [156], do not wait until the end of an episode, but they update the value function estimates after each action. This has the advantage that learning already takes place during interaction with the environment. However, immediate rewards only have a weak effect on earlier states, and many iterations are required to propagate delayed rewards back to the states and actions that deserve the credit or blame. In other words, the temporal credit-assignment problem is solved at the expense of sample efficiency. Moreover, acting greedily with respect to a non-optimal value function leads to insufficient exploration. Therefore, the exploration-exploitation dilemma is usually solved either by optimizing a stochastic policy with $\pi(a|s) > 0$ for all states and actions, or by optimizing a deterministic policy while behaving according to a different stochastic policy (*off-policy learning*).

**Policy Gradient and Actor-Critic Methods** In contrast to value-based approaches, *policy gradient methods* do not iteratively assign credit to state-action pairs in order to learn a value funtion. Instead, they directly learn a parameterized policy with parameters $\omega \in \mathbb{R}^n$ by optimizing some objective function $J(\omega)$ that represents the expected return. To maximize $J(\omega)$, policy gradient methods use *gradient ascent*:

$$\omega_{t+1} = \omega_t + \alpha \nabla_{\omega_t} J(\omega_t), \tag{6.1}$$

where $\alpha$ is the learning rate.

According to the *policy gradient theorem* [157], the gradient of the objective function can be related with the gradient of the parameterized policy as follows:

$$\nabla_\omega J(\omega) \propto \mathbb{E}_\pi \left[ q_\pi(s,a) \frac{\nabla_\omega \pi(a|s)}{\pi(a|s)} \right] \tag{6.2}$$

Taking the return $G_t$ as a sample of $q_\pi(s,a)$ results in a simple policy gradient algorithm referred to as *REINFORCE* [158] with the following update rule:

$$\omega_{t+1} = \omega_t + \alpha G_t \frac{\nabla_{\omega_t} \pi(a|s)}{\pi(a|s)} \tag{6.3}$$

Like all gradient-based optimization methods, REINFORCE can get stuck in a local optimum leaving large regions of the policy space unexplored. Furthermore, the return can have high variance which makes the algorithm rather sample-inefficient. Therefore, *actor-critic methods* (e.g., [159]) approximate $q_\pi(s,a)$ using a value-based technique so as to get the best of both worlds.

### 6.1.2 Evolution Strategies

One way to think of reinforcement learning is as an instance of (stochastic) optimization. Hence, in principle, any optimizer can be used to maximize the expected return. While policy gradient methods are, as the name suggests, *gradient-based*, it is also possible to use *gradient-free* optimizers. Not using derivative information has two advantages. First, gradient-free methods are in many cases not restricted to finding local optima. Second, the gradient of the objective function does not have to be easy to compute or even exist, so both the objective function and the policy parameterization can be chosen arbitrarily. Moreover, gradient-free policy search has been shown to be more robust than gradient-based methods regarding initialization, choice of hyperparameters, and noise [160].
Among the most successful gradient-free optimizers are the *cross-entropy method* [161], *simulated annealing* [162], and *evolutionary algorithms* [163] because they do not assume any particular knowledge on the structure of the objective function (which is sometimes referred to as black-box optimization). Evolutionary algorithms are inspired by principles of biological evolution, in particular *mutation*, *selection*, and *recombination*. We consider a subclass of evolutionary algorithms called *evolution strategies* [164] in which the search space is a subspace of $\mathbb{R}^n$, i.e., like in policy gradient methods, the goal is to find a parameter vector $\omega \in \mathbb{R}^n$ that represents an optimal policy. Since evolution strategies do not necessarily consider MDPs, the main problem to be solved is no longer credit assignment to the right states and actions but to combinations of policy parameter values.

**CMA-ES** The *Covariance Matrix Adaptation Evolution Strategy (CMA-ES)* [165] is a general purpose optimizer that has become a quasi-standard in several areas of robotics. In fact, in the present thesis, we use it for such different optimization tasks as point cloud fitting (Section 6.2.1) and policy search (Section 6.3). Like most evolution strategies, CMA-ES is a generation-based algorithm. At the beginning of each generation, $\lambda = 4 + \lfloor 3 \ln(n) \rfloor$ parameter vectors are sampled from a multivariate normal distribution centered around the current policy parameters (mutation). After that, the robot evaluates the objective function for all samples by following the respective policies. Then, the $\mu = \lfloor \frac{\lambda}{2} \rfloor$ best samples are picked out (selection), and their weighted mean is considered the new optimal policy (recombination). Furthermore, after each generation, the covariance matrix encoding the pairwise dependencies between the parameters, and a step size variable $\sigma$ balancing exploration and exploitation, are updated using paths of the policy evolution over time. The only free parameter of CMA-ES is the initial step size $\sigma_0$.

**Surrogate-enhanced Active-CMA-ES** In our experiments, we use the more sample-efficient *Active-CMA-ES* variant of the CMA-ES algorithm which introduces a negative update of the covariance matrix so that not only the best, but also the worst episodes are considered [166]. Besides, it has been shown that even less samples are required if previous samples are used to approximate the landscape of the objective function by means of a surrogate model. In some sense, the idea is similar to that of actor-critic methods in that not only the return of the latest samples but also a long term approximation of the expected return is used in order to improve sample efficiency. According to [167], surrogate models can be exploited at various stages of evolutionary optimization, and all kinds of approximation techniques have been described in the literature. We employ a simple *k-nearest-neighbor regression* model which can be used in different ways including the following two: (i) Before each generation, $\lambda_{pre} > \lambda$ samples are drawn from the distribution, and only the $\lambda$ best samples according to the model are preselected for re-evaluation by the robot. (ii) A certain proportion of real-world episodes per generation is replaced by surrogate function evaluations.
In summary, the main reasons we have preferred a surrogate-enhanced Active-CMA-ES approach to other policy search methods in our example problem are: (i) The only ingredients required are an arbitrary parameterization of the robot's policy and an objective function quantifying the final outcome of an episode. This is particularly beneficial if it is difficult to obtain reliable information about the true state of the environment and the current performance of the robot during task execution. (ii) The CMA-ES algorithm solves

the exploration-exploitation problem in a sophisticated manner, adapting the critical strategy parameters automatically. (iii) Using the enhancements of Active-CMA-ES and a surrogate model, the method can be made reasonably sample-efficient.

## 6.2 An Exemplary Dressing Task: Putting On a Knit Cap

The general dressing assistance problem is extremely challenging because of the hardly predictable dynamics of the involved objects, together with the difficulty to control the complex interactions between a garment and the body part to be dressed. Therefore, we believe that it is important to consider methods which avoid the need to model the item of clothing and its interplay with the other physical entities in full detail. In this spirit, our approaches to detection and tracking have been focused on reduced topological representations of clothes rather than exhaustive geometric models. However, occlusion and self-occlusion complicate tracking during contact-rich interactions. This motivates the work described in this chapter which explores an extreme strategy that neglects any information about the configuration of the garment *during* task execution and instead uses reinforcement learning to optimize an objective function that is based solely on information about the task performance gathered *after* an episode of the task.

We focus on the specific scenario of a robot putting a knit cap on a head. This exemplary task involves many of the aforementioned challenges that characterize robot-assisted dressing. As a simplification, we use a styrofoam head model firmly attached to a wooden base. This bypasses most of the safety issues and additional difficulties that would arise from reactive movements of a real human head. It also simplifies systematic studies of the learning approach which relies on repeated interaction with the environment.

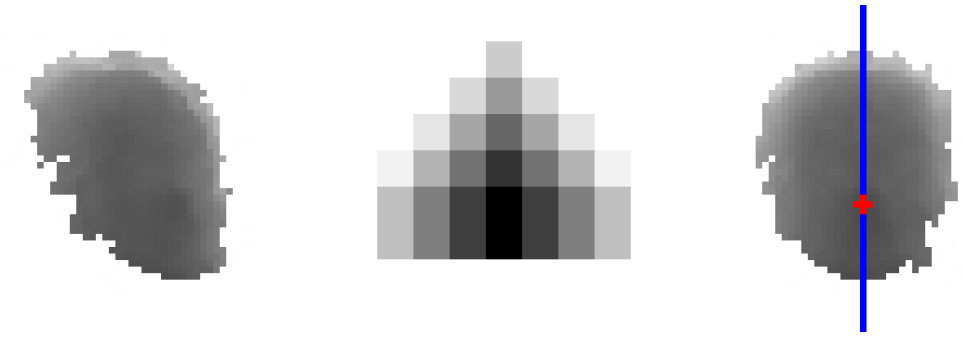
Nevertheless, the task remains very challenging because the robot is neither provided with a model of the garment nor with the full geometry of the head. Moreover, the robot receives no visual or force feedback during task execution and has to coordinate two kinematically redundant arms with attached five-fingered hands. For the movement policies to be meaningful in the context of the dressing task, we assume that the robot is initially in a certain configuration and grasps the garment in a defined way. The policy space is then a space of robot trajectories expressed in a reference frame that is based on an ellipsoidal model of the head. Hence, a robot vision algorithm is required that provides an estimate of the head pose and scale.

### 6.2.1 Ellipsoidal Model of the Head

The method described in the following provides the robot with a geometric representation of the part of the head that is to be covered by the knit cap. We emphasize that we require an estimate not only of the pose of the head but also of the scale parameters. The reader is referred to [168] for a survey of computer vision methods for human head pose estimation. Many of the existing approaches are texture-based. The styrofoam head in our example does not have much texture though. Therefore, we rely on techniques based on 3D structures which can be extracted from depth data.
We model the styrofoam head as an ellipsoid which we will use for a policy parameterization based on spherical coordinates (Section 6.2.3). Our algorithm aims to find the ellipsoid whose upper hemiellipsoid best fits the upper part of the (hairless) head. We propose a point cloud based single-view approach which is (i) independent of texture, shadows, and skin color; and (ii) designed to be used in a realistic robotic setup including a single depth camera with a downward diagonal viewing direction.
In general, an ellipsoid is defined through nine parameters: three for position, three for orientation, and three for scaling. Throughout this chapter, $C = (C_x, C_y, C_z)^T$ denotes the ellipsoid center. The axes $u, v, w$ of the ellipsoid object (depicted as red, green, and blue lines in Figures 6.2 and 6.3) are represented by unit vectors $\vec{u}$, $\vec{v}$, $\vec{w}$ and lengths $a$, $b$, $c$. The angles of rotation w.r.t. an extrinsic coordinate system about the intrinsic axes are denoted by $\alpha$ (about $u$), $\beta$ (about $v$), and $\gamma$ (about $w$) and applied in reverse order. The difficulty with the single-view setup is that the point cloud of the head is incomplete due to self-occlusion, so that there is no unique solution to the ellipsoid fitting problem. We employ a cascade of heuristics to initialize an ellipsoid whose parameters are then optimized to fit the point cloud.

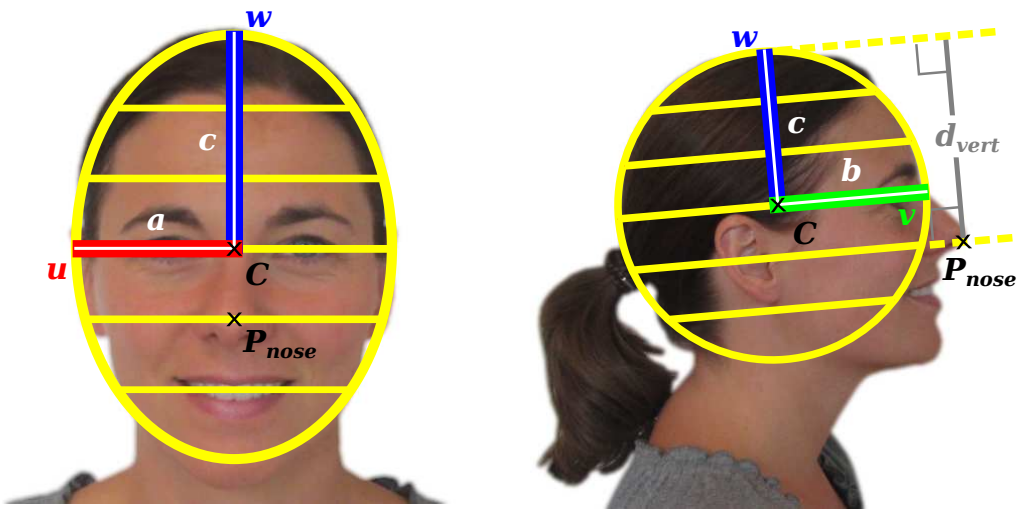
#### Ellipsoid Initialization Using Landmark Structures and Head Proportions

One way to resolve head pose ambiguities is to detect landmark structures of the face. However, there are often only two stable 3D features: the tip of the nose and the symmetry plane through the nose. We make the assumption that somewhat more than half of the face is visible to the depth camera. Then, we can use the obtained 3D point cloud to detect both features. In order to do so, we basically follow Spreeuwers' approach [169].
First, we extract those points which belong to the head by detecting the biggest connected region of points in a predefined area. As a second step, we create a set of range images (Figure 6.1(a)) by projecting the reduced point cloud to a vertical plane which is rotated in steps of one degree about the vertical axis through the center of mass of the points. To find the symmetry plane,



(a) (b) (c)

Figure 6.1: Detection of the symmetry plane of the face and the tip of the nose using point cloud data. (a) Example range image. (b) Nose template. (c) Optimal axis of symmetry (blue) and nose tip (red).



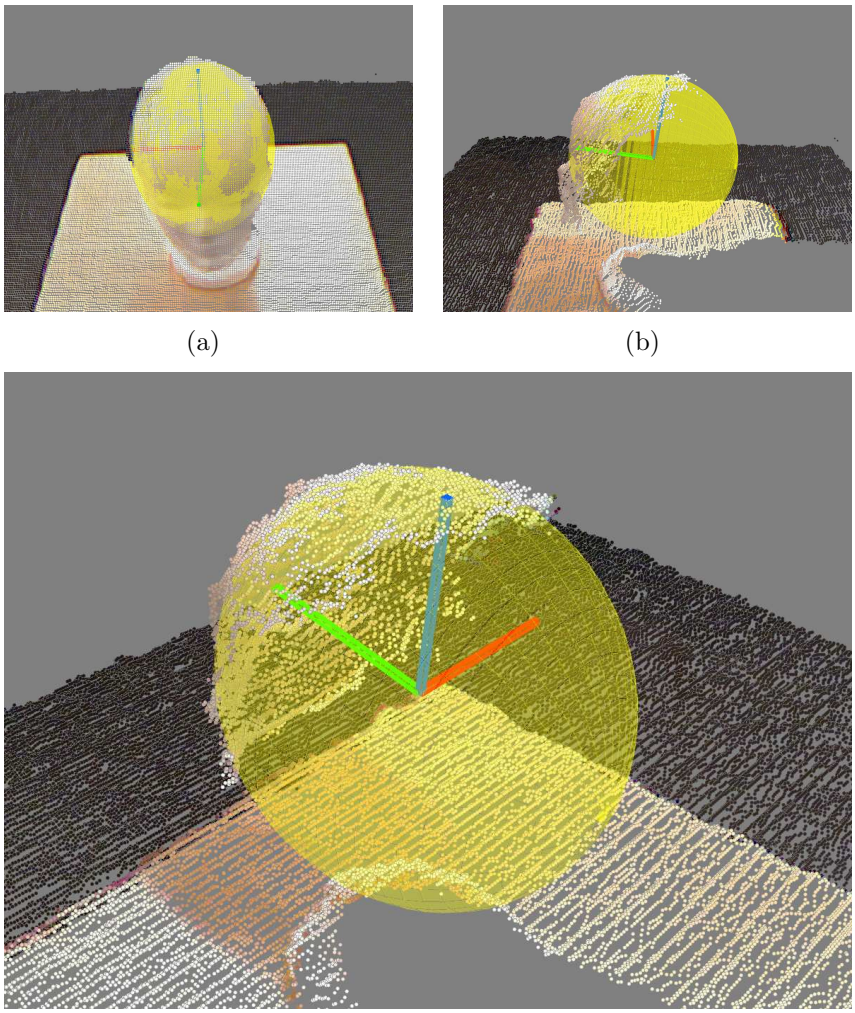

(a) (b)

Figure 6.2: Head proportion heuristics used for initialization of the ellipsoid model parameters. (a) Front view. (b) Side view.

we calculate a symmetry measure for each rotation and each shift along the horizontal axis of the range image (for details, refer to [169]). Then, for all local minima of the symmetry measure, *Normalized Cross Correlation* based nose template matching (Figure 6.1(b)) is performed along the vertical mirror axis to find an optimal symmetry plane / nose tip pair (Figure 6.1(c)).
We are now able to initialize the model parameter tuple $(C_x, C_y, C_z, \alpha, \beta, \gamma, a, b, c)$ using several heuristics. We set $\gamma$ to the optimal rotation of the symmetry plane. It can be assumed that the styrofoam head is not tilted to the side, i.e., there is no rotation $\beta$. To estimate the remaining orientation parameter $\alpha$, we employ another technique from [169]. We fit a cylinder with a fixed radius to the points within a defined region around the nose. To this end, we vary both the shift of the cylinder along $v$ and the rotation about $u$, and set $\alpha$ to the rotation angle that minimizes the mean squared distance of the points from the cylinder.
To obtain initial values for the position and scale parameters, we consider the head proportion heuristics illustrated in Figure 6.2. We see that the head can be modeled as an oblate spheroid, i.e., $b = c$. Let $P_{nose}$ be the position of the tip of the nose. We define a plane spanned by $\vec{u}$ and $\vec{v}$ and going through $P_{nose}$. Then, we compute the distances of the head points above the nose tip from that plane, define $d_{vert}$ to be a high percentile of the distances, and set $b = c = \frac{3}{4} d_{vert}$. The scale parameter $a$ is set to a high percentile of all horizontal distances of the head points from the symmetry plane, and the center of the ellipsoid is initialized as follows:

$$C = P_{nose} - b\vec{v} + \frac{1}{3} c\vec{w} \tag{6.4}$$

**Parameter Optimization Through Point Cloud Fitting** We must be careful which parameters to optimize when fitting the ellipsoid model to a noisy and incomplete point cloud. The back part of the head, for instance, is usually not represented, so we rely on the heuristic estimates of the scale parameters $b$ and $c$, and only optimize $a$. As mentioned above, we assume that $\beta = 0$. The rotation angle $\alpha$ is used to define the coordinate frame of the ellipsoid object, but rotating about $u$ does not change the shape of the oblate spheroid model. Therefore, we can only improve $\gamma$. However, the tip of the nose should be on the symmetry plane, so we optimize a rotation angle $\gamma'$ about an axis $w'$ with direction $\vec{w}$ and going through $P_{nose}$ instead. Moreover, we do not optimize the global position $C$ of the ellipsoid center, but two parameters $C_v$ and $C_w$ representing translation along this new symmetry plane.
The parameter tuple $(a, \gamma', C_v, C_w)$ is optimized using Active-CMA-ES (Section 6.1.2). As objective function, we use the mean squared distance of the



(a) (b)

(c)

Figure 6.3: Ellipsoid model fitted to the noisy and incomplete point cloud of a styrofoam head. The intrinsic coordinate frame is depicted as red (u-axis of length a), green (v-axis of length b), and blue lines (w-axis of length c). (a) Front view. (b) Side view. (c) Diagonal back view.

upper head points (those lying above the old $uv$-plane) from the ellipsoid. The distances are computed iteratively using the algorithm from [170] because no closed-form solution is known to the point-ellipsoid distance problem. Figure 6.3 shows the fitted ellipsoid.

**Evaluation** The experimental setup used for evaluating the ellipsoid fitting method included a styrofoam head placed on a table and a Kinect depth camera with a downward diagonal viewing angle of about 45° w.r.t. the tabletop. This was intended to simulate the situation where a human sits on a chair with the face roughly directed toward the robot. To test the performance of our algorithm in multiple configurations, we randomly placed the head on the tabletop at ten slightly different positions and horizontal orientations (within a range of about ±10 cm and ±20° as measured from the vertical plane along the viewing direction of the camera). In doing so, we varied the structure and amount of self-occlusion of the head w.r.t. the depth camera. We were interested (i) in how much these variations affected the repeatability (standard deviation) of the estimation of the other model parameters (the rotation angle $\alpha$ about $u$ and the scale parameters), and (ii) in how well the model fitted the visible and invisible parts of the upper head.
From visual inspection, we found that the model fitting was reasonable in all trials. The standard deviation of $\alpha$ was 0.95°. The repeatability of the estimation of $a$ (SD = 0.64 mm) was better than that of the other scale parameter $b = c$ which had a standard deviation of 2.3 mm. Figure 6.4 shows the evolution of the root mean squared distance (RMSD) of the visible upper head points from the ellipsoid model, averaged over the ten trials. After 30 Active-CMA-ES generations, it has converged to a value of 2.61 mm (SD = 0.27 mm).
To investigate the accuracy of the model w.r.t. both the visible and invisible parts of the head, we created a baseline mesh model of the styrofoam head in advance employing a high-precision laser scanner[1] (Figure 6.5). Then, we used ICP [143] to align the mesh (including the wooden base) with the point cloud and measured the distances of the upper head vertices from the ellipsoid model. The RMSD averaged over the ten trials was 3.6 mm (SD = 0.75 mm).

### 6.2.2 Head-centric State Space

When defining the head-centric state space, we assume a certain garment pose relative to the robot, but we do not model it explicitly. Specifically, the fingers of both hands are assumed to grasp around the boundary of the fabric with

[1] We would like to thank Jascha Achenbach for creating the baseline mesh model.

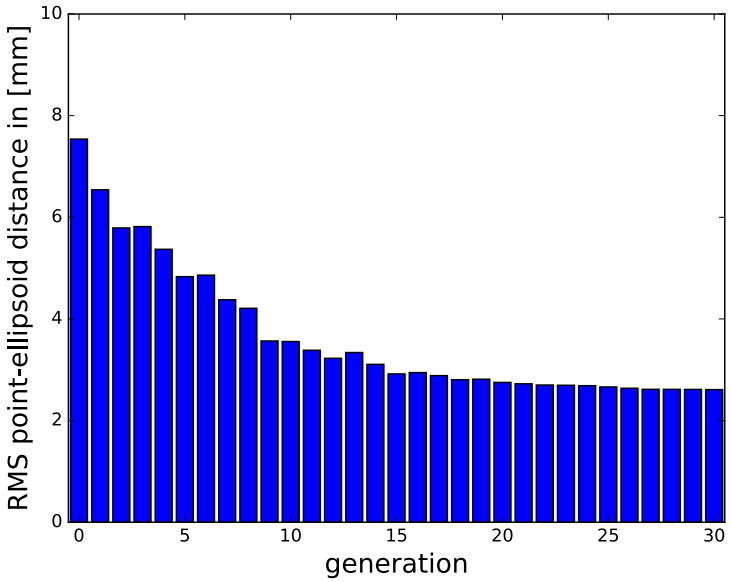
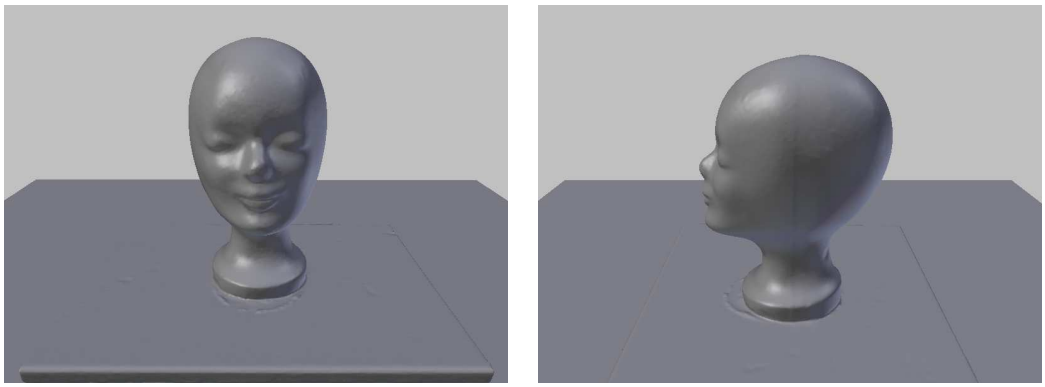



Figure 6.4: Root mean squared distance (RMSD) of the upper head points from the ellipsoid model, averaged over ten trials. Each generation comprises $\lambda = 8$ objective function evaluations.



(a) (b)

Figure 6.5: Baseline mesh model of the styrofoam head used in our experiments. (a) Front view. (b) Side view.

the thumbs pointing inside the knit cap. The robot has to infer strategies to bring the garment into the desired configuration from interaction with the environment. Therefore, the state space should (i) be restricted to only allow robot poses which are safe for the robot arms and hands, the garment, and the head; and (ii) allow fast learning, e.g., by keeping the dimensionality low.

**Hand Positions** We encode the end effector position (the contact point of the thumb and the forefinger) in spherical coordinates $(\theta, \phi, \Delta)$ w.r.t. the upper hemiellipsoid of the head model. The polar coordinate $\theta$ specifies the angle between $-\vec{v}$ and the position vector of the end effector w.r.t. $C$. The azimuth coordinate $\phi$ defines the angle between the position vector's orthogonal projection on the $uw$-plane, and $-\vec{u}$ or $+\vec{u}$ (such that $\phi \leq \frac{\pi}{2}$). To put it less formally, the polar angle $\theta \in [0, \pi]$ runs from the back pole to the front pole of the ellipsoid, whereas the azimuth angle $\phi \in [0, \frac{\pi}{2}]$ determines the "meridian" on which the end effector is located at a given $\theta$. Both coordinates are stretched according to the values of $a$, $b$, and $c$. The third coordinate $\Delta$ represents the distance of the finger tips from the ellipsoid model.
To obtain the cartesian coordinates $Q$ of the end effector, we employ

$$\begin{aligned} Q = C &\pm (a + \Delta) sin(\theta) cos(\phi) \vec{u} \\ &- (b + \Delta) cos(\theta) \vec{v} \\ &+ (c + \Delta) sin(\theta) sin(\phi) \vec{w}. \end{aligned} \tag{6.5}$$

The $\pm$ operator indicates the symmetry of the left and right hand w.r.t. the $vw$-axis. Restricting the hand motions to be symmetric halves the dimensionality of the state space. We enforce a minimum distance of 5 cm between the hands to prevent collisions. Besides, we note that adding $\Delta$ to the scale parameters does not result in a strictly constant distance between $Q$ and the ellipsoid surface, but is a reasonable approximation if the values of $a$, $b$, and $c$ do not differ too much.

**Hand Orientations** The hand orientations are predefined for given positions to ensure safe poses and motions. In robotic grasping, end effector orientations are often specified by two unit vectors: an approach vector, and an orientation vector (Section 4.3.1). Now, we follow a similar convention, replacing the approach vector by a pull vector. The position $Q$, the orientation vector $\vec{o}$, the pull vector $\vec{p}$, and the normal vector $\vec{n} = \vec{o} \times \vec{p}$ form the intrinsic coordinate frame of the hand (Figure 6.6).
A safe end effector orientation can be achieved by choosing the vectors $\vec{o}$ and $\vec{p}$ in such a way that they span a tangent plane to the head ellipsoid, with $\vec{p}$

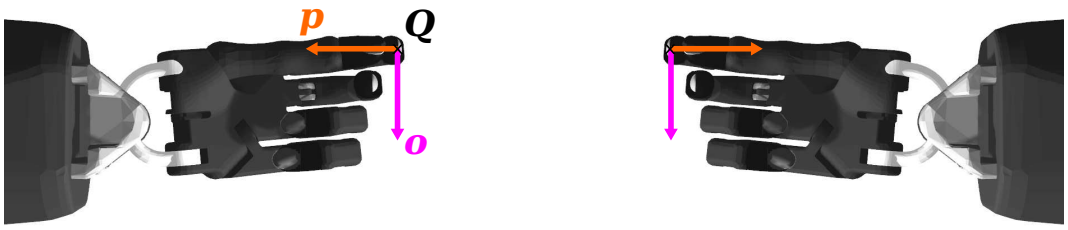



Figure 6.6: Hand frames used during the knit cap dressing experiments.

being oriented along the "meridian" through $Q$ (one of the lines running from the back pole to the front pole). This ensures that (i) the robot hands do not collide with the head (as the palms are roughly tangential to the head), and (ii) when moving the hands from the back to the front of the head, the fingers do not get caught in the fabric, but, in case of too much external resistance, the cap slips out between the thumb and the forefinger in a controlled (orthogonal) way.
The described hand orientations correspond to the following normalized gradients:

$$\begin{aligned}\vec{o} = -\frac{\nabla_\phi Q}{|\nabla_\phi Q|} = \Big[ &\pm (a+\Delta) sin(\theta) sin(\phi) \vec{u} \\ &- (c+\Delta) sin(\theta) cos(\phi) \vec{w} \Big] \Big/ |\nabla_\phi Q|\end{aligned} \tag{6.6}$$

$$\begin{aligned}\vec{p} = \frac{\nabla_\theta Q}{|\nabla_\theta Q|} = \Big[ &\pm (a+\Delta) cos(\theta) cos(\phi) \vec{u} \\ &+ (b+\Delta) sin(\theta) \vec{v} \\ &+ (c+\Delta) cos(\theta) sin(\phi) \vec{w} \Big] \Big/ |\nabla_\theta Q|\end{aligned} \tag{6.7}$$

The vectors $\vec{o}$ and $\vec{p}$ can be made perfectly orthogonal by rotating them in opposite directions about $\vec{n}$. In practice, the elbows or other parts of the robot arms would in some cases exceed the workspace limits to reach the proposed end effector poses. Therefore, we have to partly give up the conditions on the hand orientations. In the back part of the head, we limit the upward angle of the pull vector, but try to keep it on the tangent plane to avoid robot-head collisions. In the front part (where robot-head collisions are not a major issue), the downward angle of the pull vector is limited in such a way that the orientation vector remains essentially unchanged. Moreover, we restrict the horizontal angle between the forearms. The details of the constraint handling procedure are however out of the scope of this thesis.

Very low values of $\Delta$ involve the risk of collisions with the head, whereas too high values might lead to excessive stretching of the fabric. Therefore, we can further reduce the dimensionality by assuming a fixed distance $\Delta = 1$ cm of the finger tips from the head model.

### 6.2.3 Policy Parameterization

While the behavior policies during CMA-ES based learning are stochastically sampled from a multivariate normal distribution, we consider a deterministic target policy to be optimized. Hence, the policy parameters should uniquely specify for each state of the robot hands (relative to the head ellipsoid) how to proceed. We predefine a suitable initial configuration (both hands positioned at the back of the head and grasping the knit cap) and target pose (both hands at the front of the head). Then, policy search reduces to finding a parameterized end effector trajectory from the back pole to the front pole of the ellipsoid in the $(\theta, \phi)$ space. We disregard paths from the front to the back of the head in the present work.
We parameterize the trajectories as *B-splines* which have been used successfully in robotics (e.g., [171]). Specifically, we use *non-periodic uniform B-Splines* $B(t)$ of degree 3 which are fully specified by $N \geq 4$ control points that determine the shape of the path. Other authors have used policy parameterizations which do not directly depend on a parameter $t$ (such as *Dynamic Movement Primitives* [172]). However, using *B-splines*, it is easy to exploit the prior knowledge that the end effector should move continuously from the back to the front of the head, by making the spline a function of the polar angle $\theta$ instead of $t$. Thus, $B(\theta)$ expresses the behavior of the azimuth angle $\phi$ w.r.t. the running parameter $\theta$. The number of control points $N$ (which equals the dimensionality $n$ of the policy parameter vector because the control points are one-dimensional) is the only hyperparameter of this parameterization.

## 6.3 Policy Search in the Example Task

After specifying the policy space, two challenges remain: (i) defining a task-specific objective function and (ii) finding suitable hyperparameters for learning. Designing the objective function is challenging as it is not obvious how the distance from a successful reference outcome can be defined. This is because failures to properly finish the execution of a planned trajectory may be frequent in the early stages of learning. The robot could try to maximize the distance covered along the path before failing. This would however favor strategies that avoid garment-head contact completely. Therefore, we use an objective func-

tion that enables the robot to learn a trade-off between establishing contact and minimizing the risk of early failure. To support hyperparameter search and a structured analysis of the optimization process without the need for time-consuming real-world samples, we use a toy problem that mirrors some of the task constraints. Finally, we present and discuss the results of our robot's attempt to find an optimal policy for the task of putting a knit cap on a styrofoam head.

### 6.3.1 Objective Function

The objective function quantifies the performance feedback the robot gets after each episode (trajectory execution). For our example task, we suggest the objective function below. The robot receives a fixed reward $r_{success}$ for putting on the knit cap successfully. In the other cases, the objective function aims to support the robot in finding a trade-off between minimizing the risk of early failure, and establishing contact between the fabric and the head. This is expressed as a linear combination of two variables: the polar angle $\theta_{fail}$ at which the robot fails (the knit cap slips out between the thumb and the forefinger), and the average azimuth $\phi_{mean}$ along the path. From evolutionary computation, we adopt the convention that the objective function is called $f$ (rather than $J$) and is to be minimized.

$$f(P_1, ..., P_N) = \begin{cases} -r_{success}, & \text{if successful} \\ f_{shaped}(P_1, ..., P_N), & \text{otherwise} \end{cases} \tag{6.8}$$

with

$$f_{shaped}(P_1, ..., P_N) = c_{contact}\phi_{mean} - (1 - c_{contact})\theta_{fail} \tag{6.9}$$

where $P_1, ..., P_N$ denote the control points of the *B-spline* policy parameterization.

The rationale behind the shaped objective function is as follows: The further down the robot hands go (i.e., the lower the azimuth angle $\phi$ is) when moving around the head, the more contact is established between the fabric of the cap and the head. This is a very rough heuristic contact estimate and could of course be replaced by a more sophisticated force-based measure. If $\phi_{mean}$ is very low, the first term becomes minimal, but the robot is likely to fail early. If $\phi$ is constantly high, the knit cap will probably drop down only when releasing it at the end of the trajectory. In this case, the second term is minimized, but the policy is outperformed by every other policy with the same outcome and a slightly lower path. The coefficient $c_{contact}$ is a hyperparameter which controls the relative importance of the objectives.

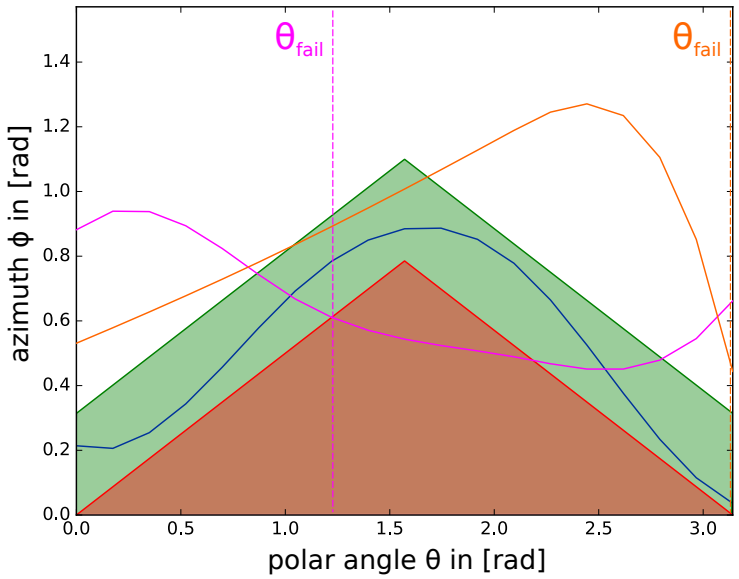



Figure 6.7: Toy problem used for hyperparameter search. The regions of successful policies and early failure are depicted in green and red, respectively. The example paths depicted as colored lines represent success (blue), early failure (pink), and failure at the end of the trajectory (orange).

### 6.3.2 A Toy Problem for Structuring Hyperparameter Search

Experiments with a robot are time-consuming and may lead to material wearout. Therefore, it is desirable to reduce the amount of real-world interaction needed for learning a task. Simulation can be useful, but tends to be overly complex, in particular in the absence of an accurate model of the involved materials. In our example, the exact physical properties of the garment, the head, and the fingertips of the robot are unknown. This is one of the reasons why we rely on real-world samples during the actual learning phase. However, hyperparameter search and an analysis of the problem structure (e.g., the influence of delayed feedback) require a particularly large number of samples. Therefore, they are performed in a simplified problem that shares some properties with the real problem but avoids costly objective function evaluations in a physical environment.

**Problem Definition and Characteristics** We have designed a toy problem analogous to the real task, i.e., a problem with the same policy parameterization, identical parameter dimensionality and range, as well as a similarly structured objective function. The exact shape of the landscape of $f$ in the real world is of course not known, but it is reasonable to assume a certain topology: There is a tube-shaped area (the green region in Figure 6.7) in the $(\theta, \phi)$ space which contains the successful policies (e.g., the path depicted in blue). Below that area, there is another area (depicted in red) that causes immediate failure when entered by a path (e.g., by the pink one). This represents the case when the knit cap slips out between the fingers because of too much contact between the cap and the head. All other policies (e.g., the path depicted in orange) fail when releasing the knit cap at the end of the trajectory because the established contact is not sufficient. We believe that the convergence behavior of the optimizer is largely determined by the overall structure of $f$, whereas the exact geometry plays a minor part. Hence, we are optimistic that hyperparameters found in the toy problem are also a good choice for the real-world problem. To be more realistic, we add a stochastic delay to $\theta_{fail}$ using Gaussian noise. This accounts for delayed sensor or user feedback, and the fact that failures are often caused by suboptimal behavior at an earlier stage of the trajectory.

**Hyperparameter Search** The toy problem was repeatedly solved to find suitable hyperparameters for the real dressing task, and to analyze the effect of delayed feedback on the optimization algorithm. The considered parameters were: $N$ (the number of control points in the *B-spline* policy parameterization), $c_{contact}$ (the weighting coefficient of the objective function), $\sigma_0$ (the initial step size used by the *Active-CMA-ES* optimizer), $k$ (the free parameter of the surrogate model), and the number of surrogate function evaluations per generation (if any). We did not perform an automated hyperparameter search because (i) the performance criterion was not clear (speed of learning, probability of convergence, or something else), and (ii) there was a risk of overfitting the hyperparameters to the simplified toy problem.

Instead, we performed a manual search and varied one parameter after the other to illustrate and discuss the choices made. We set $r_{success} = 10$ and conducted 1000 sessions (i.e., learning cycles from initialization until convergence) per hyperparameter variation. At each session, the stopping criterion for the optimizer was whether the current optimal policy was within the defined success area (green in Figure 6.7). In Figure 6.8, we show the amount of successfully finished sessions after a given number of episodes. We introduced

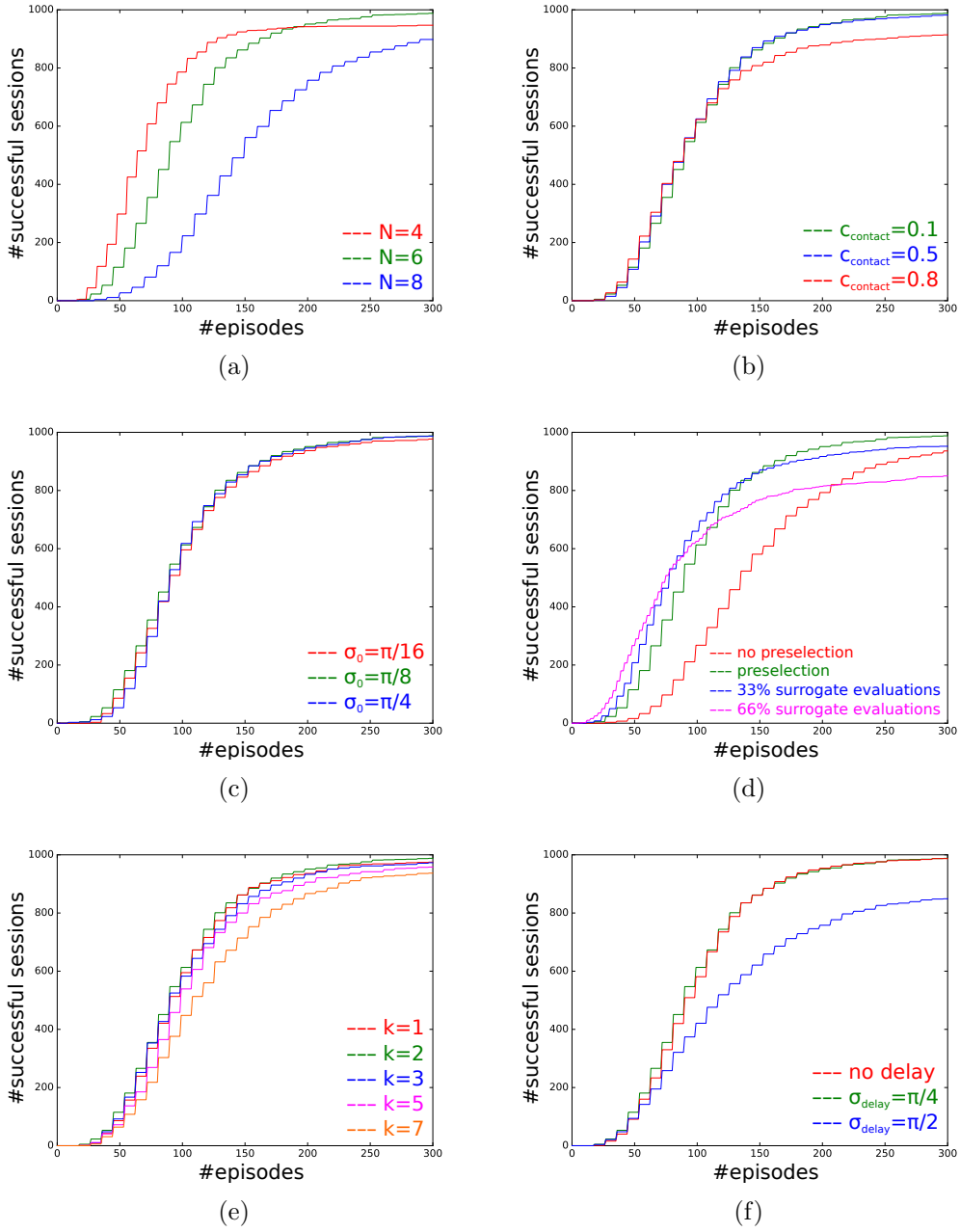


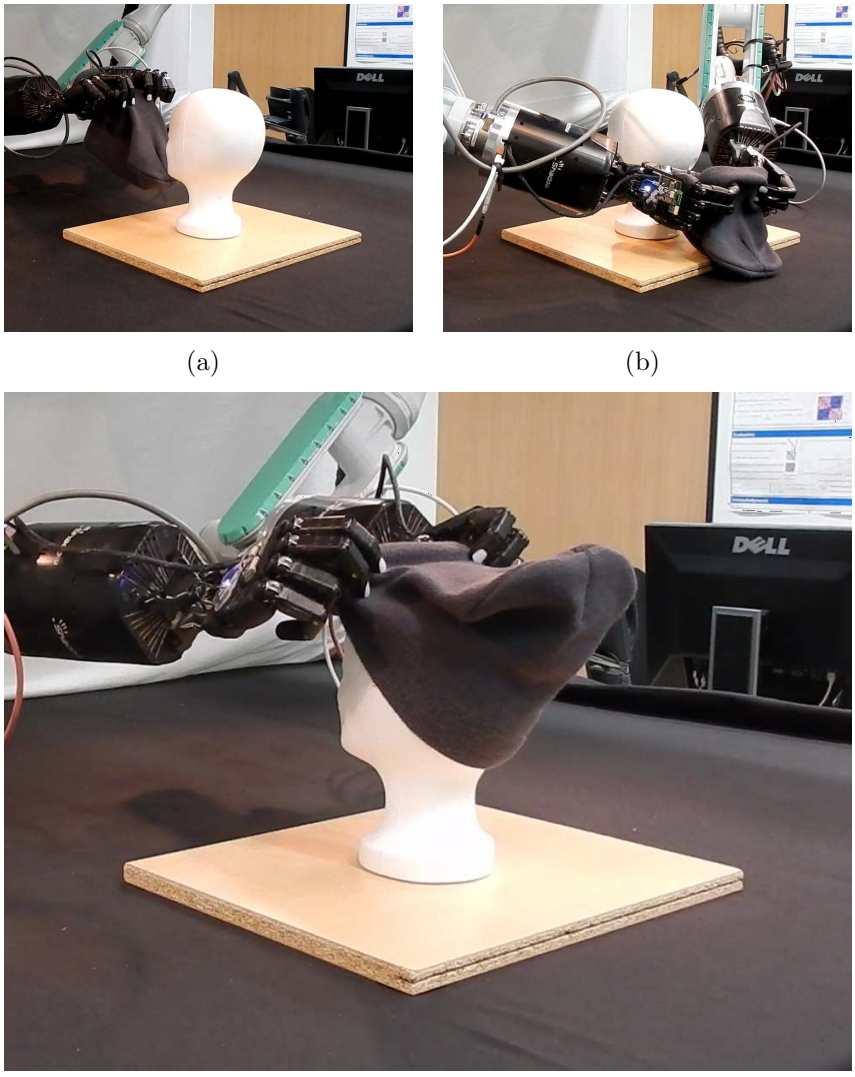

Figure 6.8: Amount of successful learning sessions after a given number of episodes under several hyperparameter variations in the toy problem.

a limit assuming that sessions which were not solved after 300 episodes had erroneously converged to a local minimum.
The choice of $N$ heavily influenced the speed of convergence. Learning in low-dimensional policy spaces was fast, but also prone to premature convergence (Figure 6.8(a)). Besides, the *B-spline* trajectory parameterization can only represent $N-2$ changes of direction, so the number of control points required to specify successful policies in the real world is not known in advance. Choosing $N = 6$ was considered a reasonable trade-off. We set $c_{contact} = 0.1$, but interestingly, the parameter had little influence on the convergence behavior, as long as $c_{contact} \leq 0.5$ (Figure 6.8(b)). As can be seen from Figure 6.8(c), the optimizer was very robust regarding $\sigma_0$. We set $\sigma_0 = \frac{\pi}{8}$. Figure 6.8(d) shows that using the surrogate model for preselection ($\lambda_{pre} = 60$) greatly speeded up learning. Replacing real objective function evaluations by surrogate evaluations drastically increased the risk of premature convergence to a local minimum, so we decided for a pure preselection strategy. The choice of $k$ in the k-nearest-neighbor regression model played a minor part, but local models (e.g., $k = 2$) performed slightly better (Figure 6.8(e)). Our method was robust to moderate Gaussion delay ($\sigma_{delay} \leq \frac{\pi}{4}$) added to $\theta_{fail}$ (Figure 6.8(f)).

### 6.3.3 Learning in the Real World

In the following, we report on the experiments we conducted with the real robot. First, we describe the experimental setup used for learning the example task. While in the toy problem, no physical objects were involved, i.e., the spherical coordinates were almost semantics-free, they now have the meaning given to them by the ellipsoid model and the head-centric policy space. A qualitative result of the experiment is that the carefully designed policy space is indeed capable of ensuring safety for the involved objects. In particular, the knit cap was not overstretched, there were no collisions between the hands and the styrofoam head, and no tendons broke in the robot fingers. The quantitative results are given below.

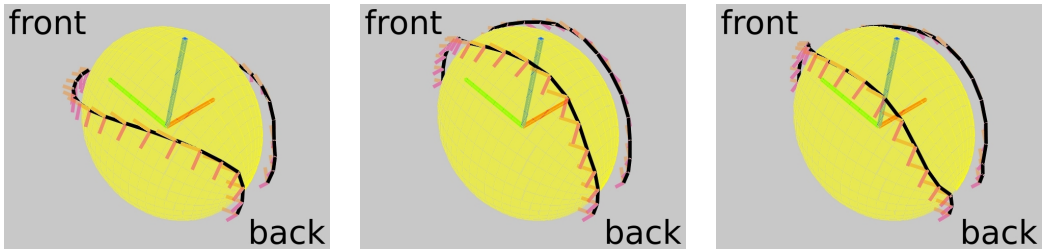
**Experimental Procedure** The robot setup in the exemplary dressing experiment consisted of two arms with attached anthropomorphic hands and a Kinect depth camera, as described in Section 1.2. Before each episode, the human experimenter placed the knit cap between the thumbs and the forefingers of the robot in such a way that the fabric covered the first two phalanges of the thumbs (Figure 6.9(a)). This step could be automated in the future using methods similar to those described in Chapter 4. The robot closed the




(a) (b)

(c)

Figure 6.9: Configurations of the three entities involved in the example task of putting a knit cap on a styrofoam head: a robot, a garment, and a head. (a) The fingers of the anthropomorphic robot hands grasping around the boundary of the fabric with the thumbs pointing inside the knit cap. (b) Starting pose of the dressing trajectory. (c) The robot executing the trajectory.




(a) (b) (c)

Figure 6.10: End effector trajectories for the task of putting a knit cap on a styrofoam head. (a) Example of a path which induces too much object-head contact. (b) Example of a path which avoids object-head contact. (c) Optimized path.

hands and moved to the starting pose at the back of the styrofoam head (Figure 6.9(b)). The CMA-ES based learning algorithm provided the parameters to be explored next, and the robot started to execute the corresponding trajectory (Figure 6.9(c)). The tendons in the fingers followed a predefined force profile protecting them against wearout [3], but no forces were applied to counteract slippage of the fabric. If the knit cap slipped out of the robot's hands, the experimenter immediately pressed a button to stop the execution. After each episode, the experimenter decided whether it was a *failed*, *successful* (the cap was placed firmly on the head; $r_{success} = 10$), or *perfect* run (the cap was placed firmly on the head, and the edge of the cap was not folded inward; $r_{success} = 20$).

**Results** As expected, two types of failure occurred during learning: (i) There was too much contact between the fabric of the cap and the head because the hands went too low (e.g., Figure 6.10(a)), and the knit cap slipped out between the thumb and the forefinger. (ii) The robot was not able to establish enough object-head contact because the path was too high (e.g., Figure 6.10(b)).
Figure 6.11 shows the average reward $(-f)$ the robot gained per generation. Learning was stopped when more than half of the episodes of one generation were *successful* or *perfect*. The objective function does not represent the different forms of deformation which occur in this region of the policy space. Therefore, adding more iterations would not help the robot learn to reliably prevent the fabric from folding inward.
After eight generations (72 episodes), the robot has learned the trade-off visualized in Figure 6.10(c). To evaluate the result, we conducted ten trials in which the robot followed the optimized policy. In half of the trials, the knit

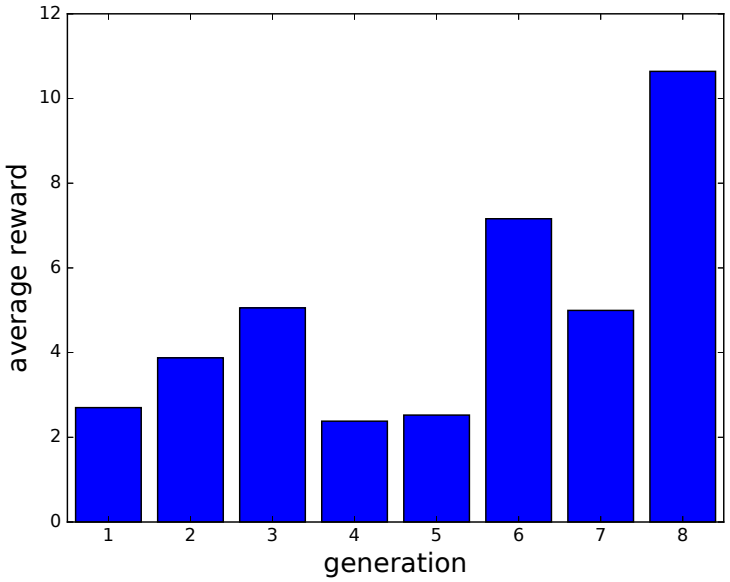
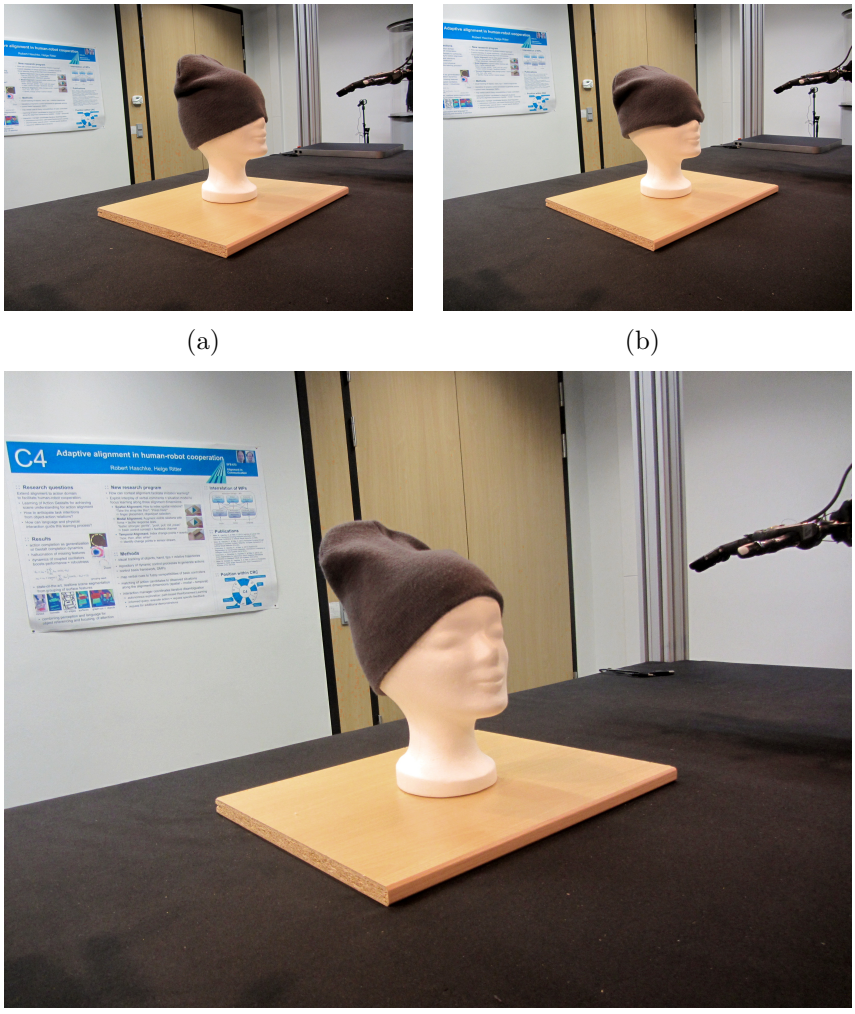



Figure 6.11: Average reward $(-f)$ gained per generation. Each generation comprises $\lambda = 9$ trajectory executions.

cap was perfectly placed on the head (Figure 6.12(a)), whereas in the other trials, parts of the edge were folded inward (Figure 6.12(b)).
A shortcoming of our method is the fact that the robot tends to pull the knit cap too far over the face of the styrofoam head. To show that this is not a major limitation, we have implemented a heuristic to uncover the eyes subsequently (Figure 6.12(c)): The robot grasps the knit cap at the highest intersection point between the fabric and the symmetry plane of the face, and pulls it backward along a line whose angle and length have been derived empirically.

## 6.4 Discussion

In this chapter, we have shown that reinforcement learning in a reduced, task-centric policy space allowed our robot to find successful trajectories for an exemplary but simplified dressing task. Although we did not model the configuration of the garment explicitly, the robot learned to put a knit cap on a styrofoam head. This can be considered an application of the interaction primitive of *pulling over*. There are several similarly structured problems both in robot-assisted dressing (e.g., learning to put on socks) and in related domains (e.g., in the task of covering a pillow).




(a) (b)

(c)

Figure 6.12: Possible configurations of the knit cap after an optimal policy rollout. (a) A *perfect* run. The cap was placed firmly on the styrofoam head. (b) A *successful* run. The cap was placed firmly on the head, but the edge was folded inward. (c) Configuration after a finalizing robot motion to uncover the eyes.

Since the general dressing assistance problem is still far from being solved, we have considered a constrained subproblem. As a consequence, our method has a number of limitations. Applying the technique to real humans would pose additional challenges such as handling heads with hair, motion tracking, and robot compliance. We were able to reduce the complexity of the problem to a tractable level, but the suggested learning approach might miss some possibly successful policies because we assume a certain grasp pose and limit the search space to paths between two fixed poles. We used a problem-specific heuristic for estimating areas of contact, and it is an open question if objective functions with similar trade-off structures can be found for other dressing tasks.
Thus far, our method does not generalize over caps or heads. However, we believe that spherical coordinates facilitate the design of algorithms with such generalization properties because they scale with the head model. For example, the robot could try to learn how a baseline trajectory must be transformed depending on the ellipsoid parameters and the size and deformability of the knit cap opening. In general, the question is how the extreme strategy of not modeling the garment at all can be softened to improve the performance of the algorithm and its ability to generalize. This could be done on the level of the objective function (e.g., by penalizing undesired folds or deviations from a baseline configuration of the garment) or on the state representation level (by including partial observations of how the garment deforms).

# 7 Conclusion

The use of clothing to cover the human body is common to all cultures worldwide. Consequently, handling garments is one of the most important everyday tasks. It is therefore not surprising that a wide variety of ideas of how technology could be useful for dealing with clothes exist in films as well as in popular and scientific literature. It has been suggested that small computing devices and motors could be integrated into the clothing material or that machines specialized in garment handling should be developed. Contrary to such approaches, we have shown in the present thesis that essential parts of the clothes perception and manipulation problem can be solved by an anthropomorphic robot in a bimanual interaction scenario. In the tasks considered, interaction occurs between various entities. Clothes interact with the robot hands, a support surface, body parts to be dressed, and other objects. Therefore, an important distinguishing feature of our work is the way these interactions are represented topologically, geometrically, and functionally. In this concluding chapter, we summarize the main results of the thesis (Section 7.1) and discuss possible directions for future research (Section 7.2).

## 7.1 Summary

To organize our thoughts, in Section 1.1, we raised six fundamental research questions related to handling clothes. After analyzing and decomposing the problem as well as developing and evaluating potential solutions using a bimanual robot, we are now able to provide informed answers which we formulate as six lessons learned in the course of the project:

*Consider the role of garment openings and contact relations.*
: To identify the similarities and dissimilarities between different tasks, we have developed a taxonomy of interaction primitives with clothes. We found that contact relation changes and the role of garment openings are reasonable categorization criteria. While there is a comparatively large body of research on garment manipulation tasks such as folding and unfolding, in our experiments, we focused on interactions that involve openings and have an effect on the contact relation between a piece of

clothing and the robot hands (in particular, *grasping* around the boundary of an opening) or another object (*hanging up* a garment, *pulling over*, *putting sth. in*, and *sliding sth. through*). Identifying garment openings and contact relations has also played an important part in our robot vision algorithms. We have presented two methods for detecting openings as well as a model for tracking. Moreover, we have shown that contact between a garment and a robot hand can be inferred indirectly from the configuration and size of the garment opening during grasping, and that the amount of contact created between the cap and the head is a good performance measure for the task of putting on a knit cap.

*Make assumptions, but do not over-constrain.*

The extreme deformability of most garments results in high-dimensional configuration spaces and very complex dynamics during manipulation and interaction with other objects. We were able to simplify many of the problems considered in this thesis by introducing some assumptions into our models. For example, we presupposed a garment lying spread out on a support surface in our polygon-based detection approach. While, depending on the scenario, this can be a reasonable assumption, we also tested the performance of the method in situations in which the condition was not satisfied, yielding limited results. In our *Active Boundary Component Models (ABCMs)* for tracking garment openings, the model assumptions have been formalized through position-based constraints. Again, we observed good performance only as long as the assumptions were generally met. We conclude that constraints are often necessary but have to be carefully considered.

*Vision is not enough.*

In our problem analysis, we identified two key challenges of perception in the clothing domain, namely to estimate the configuration of individual garments and to detect spatial relations between different objects. In our experiments, we focused on RGB-D vision and were able to show that this is sometimes sufficient for solving these two problems. However, we also demonstrated that robotic interaction is often required to render visual perception of relevant structures possible (*interactive perception*). For example, we suggested that a garment should be lifted from the support surface first to make the openings visible. Besides, it has become clear that robust occlusion handling during garment tracking and more accurate contact estimation in robot-assisted dressing are hardly possible without integrating additional sensory modalities into our models.

*Spatial representations should combine concepts from topology and geometry.*
In recent years, a few researchers have used concepts from topology to develop qualitative object and task representations for robots. We believe that such representations are particularly meaningful in interaction with clothes because the topological invariants are just those properties that remain unchanged under the most typical garment deformations and hence correlate strongly with the affordances of clothing. Therefore, we conducted a detailed topological analysis of clothes, revealing that the openings (more formally, the boundary components) and their interconnections (represented as skeletons) characterize garments not only functionally but also topologically. The applicability of topological invariants alone is limited, but the boundary component and skeleton models that have been based on this analysis proved to be useful in interaction tasks with clothes when they were combined with geometric concepts such as position, length, convexity, smoothness, planarity, and entanglement.

*State and action representations do not have to be task-agnostic.*
One of the basic motivations behind our work has been the idea that one day anthropomorphic general purpose robots might assist humans in various everyday situations. Many roboticists would probably associate this with another related idea, namely that of general purpose artificial intelligence. However, it is also possible that future robots use standardized hardware but still maintain different representations for different tasks. As far as interaction with clothes is concerned, we have presented some arguments for using task-centric state and action representations rather than working with raw sensory data and low-level motor commands. Our experimental results suggest that such representations allow robots to solve complex problems including tasks from the dressing assistance domain.

*Use as much prior knowledge as available, learn as much as necessary.*
In several experiments, we demonstrated how domain knowledge about clothing can be exploited by a robot. To this end, we integrated different types of prior knowledge into our models and algorithms. Topological prior knowledge includes the number of openings associated with a given type of clothing as well as the fact that most (but not all) garments can be considered as consisting of a single piece of fabric and hence are aptly represented as connected manifolds. Geometric prior knowledge has been provided, e.g., in the form of constraint functions or in the form of polygonal templates representing prototypical shapes of spread-out garments. The point cloud filter we have developed makes use of semantic knowl-

edge about task-relevant and irrelevant objects in the scene. Moreover, functional prior knowledge about the inherent purpose of garments has played an important role in the specification of the dressing task. In general, we think that explicit knowledge should be used if available, but learning is necessary to acquire contact-rich interaction skills such as pulling a garment tightly over another object.

## 7.2 Recommendations for Future Research

There are various possible directions for follow-up research to the work presented in this thesis. One interesting direction would be to consider robotic interactions with clothes in which our current assumptions do not hold. Thus far, we have only regarded topology-preserving interaction primitives, but in many everyday situations, it is necessary to change the topology of a garment by opening or closing it (e.g., using a zipper or buttons). Turning a piece of clothing inside out is another useful skill that is not readily expressible using our models because they assume that the orientation of the garment does not change during manipulation so that there is no need to maintain a representation of the inner surface (which in this case would be folded outward).

In large parts of this thesis, we limited ourselves to algorithms for robot vision, but we think that, in future work, different sensory modalities should be integrated. For example, tactile sensors in the fingertips could be used to deal with visual occlusion during detection and tracking of garment openings. To avoid distorting physical contact, short-range distance sensors might be employed instead. Moreover, proprioception could serve as a prior during tracking since a point on the garment surface close to where the robot holds the garment will also be close to the grasping point after moving the arm. Force/torque sensing while pulling at an opening could be used to infer the stretchability of the material and/or the size of the opening. To obtain a better contact estimate between the garment and a body part to be dressed, direct pressure measurement, e.g., in the styrofoam head during knit cap dressing, is a possibility in artificial learning scenarios in the lab. By contrast, indirect force measurement at the robot's end effector is more practical in real-world applications.

To move from lab studies to field studies, it is also necessary to consider human factors such as reactive movements during robot-assisted dressing. Furthermore, humans might act as teachers providing expert knowledge and giving feedback, or as cooperators in everyday tasks such as doing the laundry. In this context, efficient task allocation and communication about clothing are relevant research topics.

Another open question is how different levels of abstraction can be optimally combined in robotic representations of garments and their interactions with other objects. As far as spatial representations are concerned, there are presumably many different ways of how high-level topological models can be built up from low-level geometric features. Concerning task representations, we believe that a lot of research is still needed before machine learning methods can be used out of the box for acquiring complex interaction skills with clothes from raw data. However, present-day deep learning architectures are suitable for recognition and other subtasks for which much data is available. It is a long-term research question how such neural networks can be organized hierarchically to make high-level skill learning possible.
We think that some of our ideas can be transferred from the clothing domain to other application areas. The perception and grasping methods could be adapted to similar deformable objects with openings such as bags or pillowcases. Besides, the algorithmic contributions from Chapter 6 (point cloud based head pose and scale estimation as well as k-nearest-neighbor enhanced CMA-ES optimization) might be useful for other tasks. Furthermore, in the knit cap dressing task, the amount of contact between two objects was maximized, but at the same time, the risk of failing immediately (slipping off) had to be minimized. We ask how objective functions can be designed for similarly structured tasks from other domains. Finally, from a basic research perspective, it would be interesting to further investigate the relationship between topology and affordances. In particular, it is an open question whether the concept of interactive perception of topological properties is applicable to a broader range of everyday objects.

# Bibliography


[1] T.-H.-L. Le, M. Jilich, A. Landini, M. Zoppi, D. Zlatanov, and R. Molfino, "On the development of a specialized flexible gripper for garment handling," *Journal of automation and control engineering*, vol. 1, no. 3, pp. 255–259, 2013.

[2] E. Simo-Serra, F. Moreno-Noguer, and A. Perez-Gracia, "Design of non-anthropomorphic robotic hands for anthropomorphic tasks," in *ASME International Design Engineering Technical Conferences & Computers and Information in Engineering Conference*, 2011, pp. 377–386.

[3] G. Walck, R. Haschke, M. Meier, and H. Ritter, "Robot self-protection by virtual actuator fatigue: Application to tendon-driven dexterous hands during grasping," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2017, pp. 2200–2205.

[4] L. Twardon and H. Ritter, "Interaction skills for a coat-check robot: identifying and handling the boundary components of clothes," in *IEEE International Conference on Robotics and Automation*, 2015, pp. 3682–3688.

[5] ——, "Active boundary component models for robotic dressing assistance," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2016, pp. 2811–2818.

[6] ——, "Learning to put on a knit cap in a head-centric policy space," *IEEE Robotics and Automation Letters*, vol. 3, no. 2, pp. 764–771, 2018.

[7] A. Newell and H. A. Simon, "Human problem solving: The state of the theory in 1970," *American Psychologist*, vol. 26, no. 2, pp. 145–159, 1971.

[8] J. J. Gibson, *The Ecological Approach to Visual Perception*. Houghton Mifflin Harcourt, 1979.

[9] W. W. Gaver, "Technology affordances," in *SIGCHI Conference on Human Factors in Computing Systems*, 1991, pp. 79–84.

[10] M. A. Armstrong, *Basic Topology*. Springer, 1983.

[11] J. L. Gross and T. W. Tucker, *Topological graph theory*. Wiley-Interscience, 1987.

[12] D. Katz and O. Brock, "Manipulating articulated objects with interactive perception," in *IEEE International Conference on Robotics and Automation*, 2008, pp. 272–277.

[13] J. Bohg, K. Hausman, B. Sankaran, O. Brock, D. Kragic, S. Schaal, and G. S. Sukhatme, "Interactive perception: Leveraging action in perception and perception in action," *IEEE Transactions on Robotics*, vol. 33, no. 6, pp. 1273–1291, 2017.

[14] J. Piaget and B. Inhelder, *La représentation de l'espace chez l'enfant*. Presses universitaires de France, 1977.

[15] I. Darke, "A review of research related to the topological primacy thesis," *Educational Studies in Mathematics*, vol. 13, no. 2, pp. 119–142, 1982.

[16] W. Schipper, "The topological primacy thesis: Genetic and didactic aspects," *Educational Studies in Mathematics*, vol. 14, no. 3, pp. 285–296, 1983.

[17] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, 2012, pp. 1097–1105.

[18] S. Levine, P. Pastor, A. Krizhevsky, and D. Quillen, "Learning hand-eye coordination for robotic grasping with large-scale data collection," in *2016 International Symposium on Experimental Robotics*, 2017, pp. 173–184.

[19] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," in *NIPS Deep Learning Workshop*, 2013.

[20] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis, "Mastering the game of go with deep neural networks and tree search," *nature*, vol. 529, no. 7587, pp. 484–489, 2016.

[21] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "Imagenet large scale visual recognition challenge," *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211–252, 2015.

[22] G. Marcus, "Deep learning: A critical appraisal," *arXiv:1801.00631 [cs.AI]*, 2018.

[23] R. Bajcsy, "Active perception," *Proceedings of the IEEE*, vol. 76, no. 8, pp. 996–1005, 1988.

[24] H. Durrant-Whyte and T. Bailey, "Simultaneous localization and mapping: part 1," *IEEE Robotics & Automation Magazine*, vol. 13, no. 2, pp. 99–110, 2006.

[25] S. Chen, Y. Li, and N. M. Kwok, "Active vision in robotic systems: A survey of recent developments," *International Journal of Robotics Research*, vol. 30, no. 11, pp. 1343–1377, 2011.

[26] P. Hebert, T. Howard, N. Hudson, J. Ma, and J. W. Burdick, "The next best touch for model-based localization," in *IEEE International Conference on Robotics and Automation*, 2013, pp. 99–106.

[27] V. Chu, I. McMahon, L. Riano, C. G. McDonald, Q. He, J. M. Perez-Tejada, M. Arrigo, T. Darrell, and K. J. Kuchenbecker, "Robotic learning of haptic adjectives through physical interaction," *Robotics and Autonomous Systems*, vol. 63, no. 3, pp. 279–292, 2015.

[28] D. Xu, G. E. Loeb, and J. A. Fishel, "Tactile identification of objects using bayesian exploration," in *IEEE International Conference on Robotics and Automation*, 2013, pp. 3056–3061.

[29] Q. Li, R. Haschke, and H. Ritter, "Learning a tool's homogeneous transformation by tactile-based interaction," in *IEEE-RAS International Conference on Humanoid Robots*, 2016, pp. 416–421.

[30] M. Meier, G. Walck, R. Haschke, and H. Ritter, "Distinguishing sliding from slipping during object pushing," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2016, pp. 5579–5584.

[31] A. Vásquez, Z. Kappassov, and V. Perdereau, "In-hand object shape identification using invariant proprioceptive signatures," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2016, pp. 965–970.

[32] R. M. Martín and O. Brock, "Online interactive perception of articulated objects with multi-level recursive estimation based on task-specific priors," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2014, pp. 2494–2501.

[33] J. Sturm, C. Stachniss, and W. Burgard, "A probabilistic framework for learning kinematic models of articulated objects," *Journal of Artificial Intelligence Research*, vol. 41, pp. 477–526, 2011.

[34] K. Hausman, S. Niekum, S. Osentoski, and G. S. Sukhatme, "Active articulation model estimation through interactive perception," in *IEEE International Conference on Robotics and Automation*, 2015, pp. 3305–3312.

[35] C. J. Tsikos and R. K. Bajcsy, "Segmentation via manipulation," *IEEE Transactions on Robotics and Automation*, vol. 7, no. 3, pp. 306–319, 1991.

[36] J. Kenney, T. Buckley, and O. Brock, "Interactive segmentation for manipulation in unstructured environments," in *IEEE International Conference on Robotics and Automation*, 2009, pp. 1377–1382.

[37] D. Schiebener, A. Ude, and T. Asfour, "Physical interaction for segmentation of unknown textured and non-textured rigid objects," in *IEEE International Conference on Robotics and Automation*, 2014, pp. 4959–4966.

[38] B. Willimon, S. Birchfield, and I. Walker, "Rigid and non-rigid classification using interactive perception," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2010, pp. 1728–1733.

[39] A. Ude, D. Omrčen, and G. Cheng, "Making object learning and recognition an active process," *International Journal of Humanoid Robotics*, vol. 5, no. 2, pp. 267–286, 2008.

[40] N. Lyubova, S. Ivaldi, and D. Filliat, "From passive to interactive object learning and recognition through self-identification on a humanoid robot," *Autonomous Robots*, vol. 40, no. 1, pp. 33–57, 2016.

[41] M. Krainin, B. Curless, and D. Fox, "Autonomous generation of complete 3d object models using next best view manipulation planning," in *IEEE International Conference on Robotics and Automation*, 2011, pp. 5031–5037.

[42] A. Tsuda, Y. Kakiuchi, S. Nozawa, R. Ueda, K. Okada, and M. Inaba, "On-line next best grasp selection for in-hand object 3d modeling with dual-arm coordination," in *IEEE International Conference on Robotics and Automation*, 2012, pp. 1799–1804.

[43] L. Sun, S. Rogers, G. Aragon-Camarasa, and J. P. Siebert, "Recognising the clothing categories from free-configuration using gaussian-process-based interactive perception," in *IEEE International Conference on Robotics and Automation*, 2016, pp. 2464–2470.

[44] B. Willimon, S. Birchfield, and I. Walker, "Classification of clothing using interactive perception," in *IEEE International Conference on Robotics and Automation*, 2011, pp. 1862–1868.

[45] B. Willimon, I. Walker, and S. Birchfield, "A new approach to clothing classification using mid-level layers," in *IEEE International Conference on Robotics and Automation*, 2013, pp. 4271–4278.

[46] B. Willimon, S. Birchfield, and I. Walker, "Model for unfolding laundry using interactive perception," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2011, pp. 4871–4876.

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

[48] N. Krüger, C. Geib, J. Piater, R. Petrick, M. Steedman, F. Wörgötter, A. Ude, T. Asfour, D. Kraft, D. Omrčen, A. Agostini, and R. Dillmann, "Object-action complexes: Grounded abstractions of sensorimotor processes," *Robotics and Autonomous Systems*, vol. 59, no. 10, pp. 740–757, 2011.

[49] E. S. Ho and T. Komura, "Character motion synthesis by topology coordinates," *Computer Graphics Forum*, vol. 28, no. 2, pp. 299–308, 2009.

[50] F. T. Pokorny, J. A. Stork, and D. Kragic, "Grasping objects with holes: A topological approach," in *IEEE International Conference on Robotics and Automation*, 2013, pp. 1100–1107.

[51] T. K. Dey, J. Sun, and Y. Wang, "Approximating loops in a shortest homology basis from point data," in *Twenty-sixth annual symposium on Computational geometry*, 2010, pp. 166–175.

[52] J. A. Stork, F. T. Pokorny, and D. Kragic, "A topology-based object representation for clasping, latching and hooking," in *IEEE-RAS International Conference on Humanoid Robots*, 2013, pp. 138–145.

[53] D. Zarubin, F. T. Pokorny, D. Song, M. Toussaint, and D. Kragic, "Topological synergies for grasp transfer," in *Hand Synergies - How to Tame the Complexity of Grapsing, Workshop, IEEE International Conference on Robotics and Automation*, 2013.

[54] F. T. Pokorny, M. Hawasly, and S. Ramamoorthy, "Multiscale topological trajectory classification with persistent homology," in *Robotics: Science and Systems*, 2014.

[55] W. J. Beksi and N. Papanikolopoulos, "Signature of topologically persistent points for 3d point cloud description," in *IEEE International Conference on Robotics and Automation*, 2018, pp. 3229–3234.

[56] E. S. Ho, T. Komura, S. Ramamoorthy, and S. Vijayakumar, "Controlling humanoid robots in topology coordinates," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2010, pp. 178–182.

[57] D. Zarubin, V. Ivan, M. Toussaint, T. Komura, and S. Vijayakumar, "Hierarchical motion planning in topological representations," in *Robotics: Science and Systems*, 2012.

[58] P. Vinayavekhin, S. Kudoh, and K. Ikeuchi, "Towards an automatic robot regrasping movement based on human demonstration using tangle topology," in *IEEE International Conference on Robotics and Automation*, 2011, pp. 3332–3339.

[59] W. Yuan, K. Hang, H. Song, D. Kragic, M. Y. Wang, and J. A. Stork, "Reinforcement learning in topology-based representation for human body movement with whole arm manipulation," *arXiv:1809.04322 [cs.RO]*, 2018.

[60] T. Tamei, T. Matsubara, A. Rai, and T. Shibata, "Reinforcement learning of clothing assistance with a dual-arm robot," in *IEEE-RAS International Conference on Humanoid Robots*, 2011, pp. 733–738.

[61] N. Koganti, J. G. Ngeo, T. Tomoya, K. Ikeda, and T. Shibata, "Cloth dynamics modeling in latent spaces and its application to robotic clothing assistance," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2015, pp. 3464–3469.

[62] S. Berretti, A. Del Bimbo, and P. Pala, "3d mesh decomposition using reeb graphs," *Image and Vision Computing*, vol. 27, no. 10, pp. 1540–1554, 2009.

[63] J. Aleotti and S. Caselli, "Part-based robot grasp planning from human demonstration," in *IEEE International Conference on Robotics and Automation*, 2011, pp. 4554–4560.

[64] J. Aleotti, V. Micelli, and S. Caselli, "Comfortable robot to human object hand-over," in *IEEE International Symposium on Robot and Human Interactive Communication*, 2012, pp. 771–776.

[65] H. Sundar, D. Silver, N. Gagvani, and S. Dickinson, "Skeleton based shape matching and retrieval," in *Shape Modeling International*, 2003, pp. 130–139.

[66] J. Shotton, A. Fitzgibbon, M. Cook, T. Sharp, M. Finocchio, R. Moore, A. Kipman, and A. Blake, "Real-time human pose recognition in parts from single depth images," in *Conference on Computer Vision and Pattern Recognition*, 2011, pp. 1297–1304.

[67] F. Blochliger, M. Fehr, M. Dymczyk, T. Schneider, and R. Siegwart, "Topomap: Topological mapping and navigation based on visual slam maps," in *IEEE International Conference on Robotics and Automation*, 2018, pp. 3818–3825.

[68] L. Sun, G. Aragon-Camarasa, S. Rogers, R. Stolkin, and J. P. Siebert, "Single-shot clothing category recognition in free-configurations with application to autonomous clothes sorting," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2017, pp. 6699–6706.

[69] J. Hu and Y. Kita, "Classification of the category of clothing item after bringing it into limited shapes," in *IEEE-RAS International Conference on Humanoid Robots*, 2015, pp. 588–594.

[70] K. Yamazaki and M. Inaba, "Clothing classification using image features derived from clothing fabrics, wrinkles and cloth overlaps," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2013, pp. 2710–2717.

[71] C. Kampouris, I. Mariolis, G. Peleka, E. Skartados, A. Kargakos, D. Triantafyllou, and S. Malassiotis, "Multi-sensorial and explorative recognition of garments and their material properties in unconstrained environ-

ment," in *IEEE International Conference on Robotics and Automation*, 2016, pp. 1656–1663.

[72] A. Gabas, E. Corona, G. Alenyà, and C. Torras, "Robot-aided cloth classification using depth information and cnns," in *International Conference on Articulated Motion and Deformable Objects*, 2016, pp. 16–23.

[73] E. Corona, G. Alenyà, A. Gabas, and C. Torras, "Active garment recognition and target grasping point detection using deep learning," *Pattern Recognition*, vol. 74, pp. 629–641, 2018.

[74] I. Mariolis, G. Peleka, A. Kargakos, and S. Malassiotis, "Pose and category recognition of highly deformable objects using deep learning," in *International Conference on Advanced Robotics*, 2015, pp. 655–662.

[75] W. Yuan, Y. Mo, S. Wang, and E. H. Adelson, "Active clothing material perception using tactile sensing and deep learning," in *IEEE International Conference on Robotics and Automation*, 2018, pp. 4842–4849.

[76] M. Cusumano-Towner, A. Singh, S. Miller, J. F. O'Brien, and P. Abbeel, "Bringing clothing into desired configurations with limited perception," in *IEEE International Conference on Robotics and Automation*, 2011, pp. 3893–3900.

[77] Y. Kita, T. Ueshiba, E. S. Neo, and N. Kita, "Clothes state recognition using 3d observed data," in *IEEE International Conference on Robotics and Automation*, 2009, pp. 1220–1225.

[78] Y. Kita, F. Kanehiro, T. Ueshiba, and N. Kita, "Clothes handling based on recognition by strategic observation," in *IEEE-RAS International Conference on Humanoid Robots*, 2011, pp. 53–58.

[79] Y. Li, C.-F. Chen, and P. K. Allen, "Recognition of deformable object category and pose," in *IEEE International Conference on Robotics and Automation*, 2014, pp. 5558–5564.

[80] Y. Li, Y. Wang, M. Case, S.-F. Chang, and P. K. Allen, "Real-time pose estimation of deformable objects using a volumetric approach," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2014, pp. 1046–1052.

[81] P. C. Wang, S. Miller, M. Fritz, T. Darrell, and P. Abbeel, "Perception for the manipulation of socks," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2011, pp. 4877–4884.

[82] A. Ramisa, G. Alenyà, F. Moreno-Noguer, and C. Torras, "Determining where to grasp cloth using depth information," in *International Conference of the Catalan Association for Artificial Intelligence*, 2011, pp. 199–207.

[83] ——, "Using depth and appearance features for informed robot grasping of highly wrinkled clothes," in *IEEE International Conference on Robotics and Automation*, 2012, pp. 1703–1708.

[84] ——, "Finddd: A fast 3d descriptor to characterize textiles for robot manipulation," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2013, pp. 824–830.

[85] K. Yamazaki, "Grasping point selection on an item of crumpled clothing based on relational shape description," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2014, pp. 3123–3128.

[86] J. Maitin-Shepard, M. Cusumano-Towner, J. Lei, and P. Abbeel, "Cloth grasp point detection based on multiple-view geometric cues with application to robotic towel folding," in *IEEE International Conference on Robotics and Automation*, 2010, pp. 2308–2315.

[87] P. Gibbons, P. Culverhouse, and G. Bugmann, "Visual identification of grasp locations on clothing for a personal robot," in *Conference Towards Autonomous Robotic Systems*, 2009, pp. 78–81.

[88] G. Alenyà, A. Ramisa, F. Moreno-Noguer, and C. Torras, "Characterization of textile grasping experiments," in *ICRA Workshop on Conditions for Replicable Experiments and Performance Comparison in Robotics Research*, 2012.

[89] K. Hamajima and M. Kakikura, "Planning strategy for task of unfolding clothes," *Robotics and Autonomous Systems*, vol. 32, no. 2-3, pp. 145–152, 2000.

[90] D. Triantafyllou, I. Mariolis, A. Kargakos, S. Malassiotis, and N. Aspragathos, "A geometric approach to robotic unfolding of garments," *Robotics and Autonomous Systems*, vol. 75, pp. 233–243, 2016.

[91] H. Yuba, S. Arnold, and K. Yamazaki, "Unfolding of a rectangular cloth based on action selection depending on recognition uncertainty," in *IEEE/SICE International Symposium on System Integration*, 2015, pp. 623–628.

[92] A. Doumanoglou, A. Kargakos, T.-K. Kim, and S. Malassiotis, "Autonomous active recognition and unfolding of clothes using random decision forests and probabilistic planning," in *IEEE International Conference on Robotics and Automation*, 2014, pp. 987–993.

[93] Y. Li, D. Xu, Y. Yue, Y. Wang, S.-F. Chang, E. Grinspun, and P. K. Allen, "Regrasping and unfolding of garments using predictive thin shell modeling," in *IEEE International Conference on Robotics and Automation*, 2015, pp. 1382–1388.

[94] J. Stria, V. Petrík, and V. Hlaváč, "Model-free approach to garments unfolding based on detection of folded layers," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2017, pp. 3274–3280.

[95] L. Sun, G. Aragon-Camarasa, S. Rogers, and J. P. Siebert, "Accurate garment surface analysis using an active stereo robot head with application to dual-arm flattening," in *IEEE International Conference on Robotics and Automation*, 2015, pp. 185–192.

[96] L. Sun, G. A. Camarasa, A. Khan, S. Rogers, and P. Siebert, "A precise method for cloth configuration parsing applied to single-arm flattening," *International Journal of Advanced Robotic Systems*, vol. 13, no. 2, 2016.

[97] D. Estevez, J. G. Victores, R. Fernandez-Fernandez, and C. Balaguer, "Robotic ironing with 3d perception and force/torque feedback in household environments," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2017, pp. 6484–6489.

[98] Y. Li, X. Hu, D. Xu, Y. Yue, E. Grinspun, and P. K. Allen, "Multi-sensor surface analysis for robotic ironing," in *IEEE International Conference on Robotics and Automation*, 2016, pp. 5670–5676.

[99] J. van den Berg, S. Miller, K. Goldberg, and P. Abbeel, "Gravity-based robotic cloth folding," *Springer Tracts in Advanced Robotics*, vol. 68, pp. 409–424, 2011.

[100] C. Bersch, B. Pitzer, and S. Kammel, "Bimanual robotic cloth manipulation for laundry folding," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2011, pp. 1413–1419.

[101] S. Miller, M. Fritz, T. Darrell, and P. Abbeel, "Parametrized shape models for clothing," in *IEEE International Conference on Robotics and Automation*, 2011, pp. 4861–4868.

[102] J. Stria, D. Průša, V. Hlaváč, L. Wagner, V. Petrík, P. Krsek, and V. Smutný, "Garment perception and its folding using a dual-arm robot," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2014, pp. 61–67.

[103] Y. Kita, F. Kanehiro, T. Ueshiba, and N. Kita, "Strategy for folding clothing on the basis of deformable models," in *International Conference Image Analysis and Recognition*, 2014, pp. 442–452.

[104] Y. Li, Y. Yue, D. Xu, E. Grinspun, and P. K. Allen, "Folding deformable objects using predictive simulation and trajectory optimization," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2015, pp. 6000–6006.

[105] V. Petrík, V. Smutný, P. Krsek, and V. Hlaváč, "Physics-based model of a rectangular garment for robotic folding," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2016, pp. 951–956.

[106] A. Colomé, A. Planells, and C. Torras, "A friction-model-based framework for reinforcement learning of robotic tasks in non-rigid environments," in *IEEE International Conference on Robotics and Automation*, 2015, pp. 5649–5654.

[107] A. Colomé, S. Foix, G. Alenyà, and C. Torras, "Reward-weighted gmm and its application to action-selection in robotized shoe dressing," in *Iberian Robotics Conference*, 2017, pp. 141–152.

[108] Y. Gao, H. J. Chang, and Y. Demiris, "Iterative path optimisation for personalised dressing assistance using vision and force information," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2016, pp. 4398–4403.

[109] ——, "User modelling for personalised dressing assistance by humanoid robots," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2015, pp. 1840–1845.

[110] G. Chance, A. Jevtić, P. Caleb-Solly, G. Alenyà, C. Torras, and S. Dogramadzi, ""Elbows out" - predictive tracking of partially occluded pose for robot-assisted dressing," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 3598–3605, 2018.

[111] F. Zhang, A. Cully, and Y. Demiris, "Personalized robot-assisted dressing using user modeling in latent spaces," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2017, pp. 3603–3610.

[112] A. Clegg, W. Yu, J. Tan, C. C. Kemp, G. Turk, and C. K. Liu, "Learning human behaviors for robot-assisted dressing," *arXiv:1709.07033 [cs.RO]*, 2017.

[113] Z. Erickson, M. Collier, A. Kapusta, and C. C. Kemp, "Tracking human pose during robot-assisted dressing using single-axis capacitive proximity sensing," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 2245–2252, 2018.

[114] A. Clegg, W. Yu, Z. Erickson, J. Tan, C. K. Liu, and G. Turk, "Learning to navigate cloth using haptics," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2017, pp. 2799–2805.

[115] A. Kapusta, W. Yu, T. Bhattacharjee, C. K. Liu, G. Turk, and C. C. Kemp, "Data-driven haptic perception for robot-assisted dressing," in *IEEE International Symposium on Robot and Human Interactive Communication*, 2016, pp. 451–458.

[116] W. Yu, A. Kapusta, J. Tan, C. C. Kemp, G. Turk, and C. K. Liu, "Haptic simulation for robot-assisted dressing," in *IEEE International Conference on Robotics and Automation*, 2017, pp. 6044–6051.

[117] Z. Erickson, A. Clegg, W. Yu, G. Turk, C. K. Liu, and C. C. Kemp, "What does the person feel? learning to infer applied forces during robot-assisted dressing," in *IEEE International Conference on Robotics and Automation*, 2017, pp. 6058–6065.

[118] Z. Erickson, H. M. Clever, G. Turk, C. K. Liu, and C. C. Kemp, "Deep haptic model predictive control for robot-assisted dressing," in *IEEE International Conference on Robotics and Automation*, 2018, pp. 4437–4444.

[119] A. Clegg, W. Yu, J. Tan, C. K. Liu, and G. Turk, "Learning to dress: Synthesizing human dressing motion via deep reinforcement learning," *ACM Transactions on Graphics*, vol. 37, no. 6, 2018.

[120] G. Chance, A. Camilleri, B. Winstone, P. Caleb-Solly, and S. Dogramadz, "An assistive robot to support dressing - strategies for planning and error handling," in *IEEE International Conference on Biomedical Robotics and Biomechatronics*, 2016, pp. 774–780.

[121] K. Yamazaki, R. Oya, K. Nagahama, K. Okada, and M. Inaba, "Bottom dressing by a life-sized humanoid robot provided failure detection and

recovery functions," in *IEEE/SICE International Symposium on System Integration*, 2014, pp. 564–570.

[122] S. D. Klee, B. Q. Ferreira, R. Silva, J. P. Costeira, F. S. Melo, and M. Veloso, "Personalized assistance for dressing users," in *International Conference on Social Robotics*, 2015, pp. 359–369.

[123] J. Han, L. Shao, D. Xu, and J. Shotton, "Enhanced computer vision with microsoft kinect sensor: A review," *IEEE Transactions on Cybernetics*, vol. 43, no. 5, pp. 1318–1334, 2013.

[124] H. Sarbolandi, D. Lefloch, and A. Kolb, "Kinect range sensing: Structured-light versus time-of-flight kinect," *Computer Vision and Image Understanding*, vol. 139, pp. 1–20, 2015.

[125] Y. Berdnikov and D. Vatolin, "Real-time depth map occlusion filling and scene background restoration for projected-pattern based depth cameras," in *International Conference on Computer Graphics and Vision*, 2011, pp. 200–203.

[126] S. Holzer, R. B. Rusu, M. Dixon, S. Gedikli, and N. Navab, "Adaptive neighborhood selection for real-time surface normal estimation from organized point cloud data using integral images," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2012, pp. 2684–2689.

[127] A. B. Carsten Rother, Vladimir Kolmogorov, "Grabcut: Interactive foreground extraction using iterated graph cuts," *ACM transactions on graphics*, vol. 23, no. 3, pp. 309–314, 2004.

[128] Y. Boykov and V. Kolmogorov, "An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 9, pp. 1124–1137, 2004.

[129] X. C. He and N. H. C. Yung, "Corner detector based on global and local curvature properties," *Optical Engineering*, vol. 47, no. 5, 2008.

[130] E. M. Arkin, L. P. Chew, D. P. Huttenlocher, K. Kedem, and J. S. B. Mitchell, "An efficiently computable metric for comparing polygonal shapes," *IEEE Transactions on Pattern Analysis & Machine Intelligence*, vol. 13, no. 3, pp. 209–216, 1991.

[131] J. Canny, "A computational approach to edge detection," *IEEE Transactions on Pattern Analysis & Machine Intelligence*, vol. 8, no. 6, pp. 679–698, 1986.

[132] A. Ückermann, R. Haschke, and H. Ritter, "Realtime 3d segmentation for human-robot interaction," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2013, pp. 2136–2143.

[133] L. Lam, S. W. Lee, and C. Y. Suen, "Thinning methodologies - a comprehensive survey," *IEEE Transactions on Pattern Analysis & Machine Intelligence*, vol. 14, no. 9, pp. 869–885, 1992.

[134] T. Y. Zhang and C. Y. Suen, "A fast parallel algorithm for thinning digital patterns," *Communications of the ACM*, vol. 27, no. 3, pp. 236–239, 1984.

[135] T. Hanser, P. Jauffret, and G. Kaufmann, "A new algorithm for exhaustive ring perception in a molecular graph," *Journal of Chemical Information and Computer Sciences*, vol. 36, no. 6, pp. 1146–1152, 1996.

[136] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[137] D. Comaniciu, R. Visvanathan, and P. Meer, "Real-time tracking of non-rigid objects using mean shift," in *IEEE Conference on Computer Vision and Pattern Recognition*, vol. 2, 2000, pp. 142–149.

[138] M. Kass, A. Witkin, and D. Terzopoulos, "Snakes: Active contour models," *International Journal for Computer Vision*, vol. 1, no. 4, pp. 321–331, 1988.

[139] M. Isard and A. Blake, "Condensation - conditional density propagation for visual tracking," *International Journal of Computer Vision*, vol. 29, no. 1, pp. 5–28, 1998.

[140] P. Li, D. Wang, L. Wang, and H. Lu, "Deep visual tracking: Review and experimental comparison," *Pattern Recognition*, vol. 76, pp. 323–338, 2018.

[141] C. Elbrechter, R. Haschke, and H. Ritter, "Folding paper with anthropomorphic robot hands using real-time physics-based modeling," in *IEEE-RAS International Conference on Humanoid Robots*, 2012, pp. 210–215.

[142] M. Schröder, C. Elbrechter, J. Maycock, R. Haschke, M. Botsch, and H. Ritter, "Real-time hand tracking with a color glove for the actuation of anthropomorphic robot hands," in *IEEE-RAS International Conference on Humanoid Robots*, 2012, pp. 262–269.

[143] P. Besl and N. D. McKay, "A method for registration of 3-d shapes," *IEEE Transactions on Pattern Analysis & Machine Intelligence*, vol. 14, no. 2, pp. 239–256, 1992.

[144] R. Plankers and P. Fua, "Articulated soft objects for multiview shape and motion capture," *IEEE Transactions on Pattern Analysis & Machine Intelligence*, vol. 25, no. 9, pp. 1182–1187, 2003.

[145] D. Hähnel, S. Thrun, and W. Burgard, "An extension of the icp algorithm for modeling nonrigid objects with mobile robots," in *International joint conference on Artificial intelligence*, vol. 3, 2003, pp. 915–920.

[146] T. Schmidt, R. A. Newcombe, and D. Fox, "Dart: Dense articulated real-time tracking," in *Robotics: Science and Systems*, vol. 2, 2014.

[147] A. Tagliasacchi, M. Schröder, A. Tkach, S. Bouaziz, M. Botsch, and M. Pauly, "Robust articulated-icp for real-time hand tracking," *Computer Graphics Forum*, vol. 34, no. 5, pp. 101–114, 2015.

[148] J. Schulman, A. Lee, J. Ho, and P. Abbeel, "Tracking deformable objects with point clouds," in *IEEE International Conference on Robotics and Automation*, 2013, pp. 1130–1137.

[149] M. Müller, B. Heidelberger, M. Hennix, and J. Ratcliff, "Position based dynamics," *Journal of Visual Communication and Image Representation*, vol. 18, no. 2, pp. 109–118, 2007.

[150] N. Umetani, R. Schmidt, and J. Stam, "Position-based elastic rods," in *ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 2014, pp. 21–30.

[151] F. B. Fuller, "The writhing number of a space curve," *Proceedings of the National Academy of Sciences*, vol. 68, no. 4, pp. 815–819, 1971.

[152] K. Klenin and J. Langowski, "Computation of writhe in modeling of supercoiled dna," *Biopolymers*, vol. 54, no. 5, pp. 307–317, 2000.

[153] R. de Vries, "Evaluating changes of writhe in computer simulations of supercoiled dna," *The Journal of chemical physics*, vol. 122, no. 6, p. 064905, 2005.

[154] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, 2nd ed. The MIT Press, 2018.

[155] C. J. C. H. Watkins and P. Dayan, "Q-learning," *Machine Learning*, vol. 8, no. 3, pp. 279–292, 1992.

[156] G. A. Rummery and M. Niranjan, "On-line q-learning using connectionist systems," University of Cambridge, Department of Engineering, Tech. Rep. CUED/F-INFENG/TR 166, 1994.

[157] R. S. Sutton, D. McAllester, S. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," in *Advances in Neural Information Processing Systems*, 2000, pp. 1057–1063.

[158] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Machine Learning*, vol. 8, no. 3, pp. 229–256, 1992.

[159] T. Degris, M. White, and R. S. Sutton, "Off-policy actor-critic," in *International Conference on Machine Learning*, 2012, pp. 179–186.

[160] V. Heidrich-Meisner and C. Igel, "Evolution strategies for direct policy search," in *International Conference on Parallel Problem Solving from Nature*, 2008, pp. 428–437.

[161] R. Y. Rubinstein and D. P. Kroese, *The Cross-Entropy Method: A Unified Approach to Combinatorial Optimization, Monte-Carlo Simulation, and Machine Learning*. Springer, 2004.

[162] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, no. 4598, pp. 671–680, 1983.

[163] D. Fogel, "An introduction to simulated evolutionary optimization," *IEEE Transactions on Neural Networks*, vol. 5, no. 1, pp. 3–14, 1994.

[164] I. Rechenberg, *Evolutionsstrategie '94*. Frommann-Holzboog, 1994.

[165] N. Hansen, S. D. Müller, and P. Koumoutsakos, "Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (cma-es)," *Evolutionary Computation*, vol. 11, no. 1, pp. 1–18, 2003.

[166] G. A. Jastrebski and D. V. Arnold, "Improving evolution strategies through active covariance matrix adaptation," in *IEEE International Conference on Evolutionary Computation*, 2006, pp. 2814–2821.

[167] Y. Jin, "Surrogate-assisted evolutionary computation: Recent advances and future challenges," *Swarm and Evolutionary Computation*, vol. 1, no. 2, pp. 61–70, 2011.

[168] E. Murphy-Chutorian and M. M. Trivedi, "Head pose estimation in computer vision: A survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 31, no. 4, pp. 607–626, 2009.

[169] L. Spreeuwers, "Fast and accurate 3d face recognition," *International Journal of Computer Vision*, vol. 93, no. 3, pp. 389–414, 2011.

[170] D. Eberly, "Distance from a point to an ellipse, an ellipsoid, or a hyperellipsoid," *Geometric Tools, LLC*, 2013.

[171] M. Ruchanurucks, S. Nakaoka, S. Kudoh, and K. Ikeuchi, "Generation of humanoid robot motions with physical constraints using hierarchical b-spline," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2005, pp. 1869–1874.

[172] S. Schaal, "Dynamic movement primitives - a framework for motor control in humans and humanoid robotics," in *Adaptive motion of animals and machines*. Springer, 2006, pp. 261–280.