KTH Computer Science and Communication

From Human to Robot Grasping

JAVIER ROMERO

Doctoral Thesis in Robotics and Computer Vision
Stockholm, Sweden 2011

TRITA CSC-A 2011-18
ISSN 1653-5723
ISRN KTH/CSC/A--11/18--SE
ISBN 978-91-7501-185-1

Kungliga Tekniska Högskolan
School of Computer Science and Communication
SE-100 44 Stockholm
SWEDEN

Akademisk avhandling som med tillstånd av Kungl Tekniska högskolan framlägges till offentlig granskning för avläggande av teknologie doktorsexamen i datalogi torsdagen den 15 december 2011 klockan 13.00 i Sal F3, Kungliga Tekniska Högskolan, Lindstedtsvägen 26, Stockholm.

Tryck: E-Print

Abstract

Imagine that a robot fetched this thesis for you from a book shelf. How do you think the robot would have been programmed? One possibility is that experienced engineers had written low level descriptions of all imaginable tasks, including grasping a small book from this particular shelf. A second option would be that the robot tried to learn how to grasp books from your shelf autonomously, resulting in hours of trial-and-error and several books on the floor.

In this thesis, we argue in favor of a third approach where you teach the robot how to grasp books from your shelf through grasping by demonstration. It is based on the idea of robots learning grasping actions by observing humans performing them. This imposes minimum requirements on the human teacher: no programming knowledge and, in this thesis, no need for special sensory devices. It also maximizes the amount of sources from which the robot can learn: any video footage showing a task performed by a human could potentially be used in the learning process. And hopefully it reduces the amount of books that end up on the floor.

This document explores the challenges involved in the creation of such a system. First, the robot should be able to understand what the teacher is doing with their hands. This means, it needs to estimate the pose of the teacher's hands by visually observing their in the absence of markers or any other input devices which could interfere with the demonstration. Second, the robot should translate the human representation acquired in terms of hand poses to its own embodiment. Since the kinematics of the robot are potentially very different from the human one, defining a similarity measure applicable to very different bodies becomes a challenge. Third, the execution of the grasp should be continuously monitored to react to inaccuracies in the robot perception or changes in the grasping scenario. While visual data can help correcting the reaching movement to the object, tactile data enables accurate adaptation of the grasp itself, thereby adjusting the robot's internal model of the scene to reality. Finally, acquiring compact models of human grasping actions can help in both perceiving human demonstrations more accurately and executing them in a more human-like manner. Moreover, modeling human grasps can provide us with insights about what makes an artificial hand design anthropomorphic, assisting the design of new robotic manipulators and hand prostheses.

All these modules try to solve particular subproblems of a grasping by demonstration system. We hope the research on these subproblems performed in this thesis will both bring us closer to our dream of a learning robot and contribute to the multiple research fields where these subproblems are coming from.

Acknowledgements

Wow, this has been a long ride. People jumping in and getting out, changes in destination, trajectory and pace. In my course towards obtaining my "research driving license", commonly known as PhD, many people have helped me discovering where to go, taught me lots of driving tricks and kept me awake in the long nights on the wheel. Sometimes I tagged along people who crossed my research path. For starters, without people buying me a car in the first place I wouldn't even be able to start this race. Thank you all. We did it. And the best of all is that we have a long road in front of us.

Contents

Chapter 1

Introduction

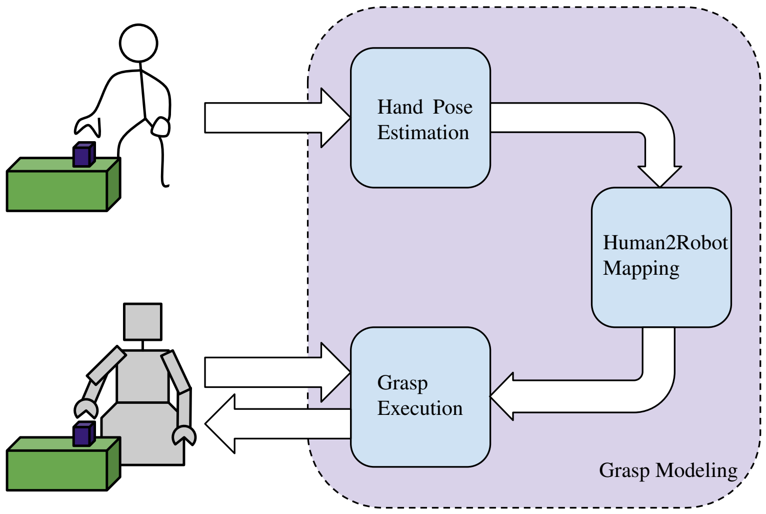


Figure 1.1: Grasping by demonstration

Look around you. Try to avoid thinking about the environment in terms of objects, actions, inter-object relations, etc. Think about which steps would be involved in the execution of an action such as getting a drink, preparing a meal or rotating this book to read the acknowledgements page. If you add to this scenario a more than imperfect control over your own body, any daily tasks such as the ones mentioned above become nearly impossible to perform. Our experience and lengthy training for tasks such as walking and manipulating objects makes us unaware of the tremendous complexity they require. Even small variations on our capabilities, such as modifications in our embodiment (e.g. a broken arm) or in our perception system (not wearing your glasses) diminish our effectiveness substantially. One important factor that reduces our capabilities to perform certain tasks is aging.

In an aging society such as ours, people require help to perform daily tasks. An attractive idea is to provide this help with robots which require low maintenance, minimum attention and salary.

However, the complexity of daily tasks differs substantially from the simplicity of the scenarios where robots have been traditionally used, e.g. industrial setups. This means that robots should be able to work in cluttered scenarios. Furthermore, the chance of inaccuracies in the execution of tasks is much higher, demanding accordingly constant checking of the state of the environment.

One of the biggest differences between the requirements for a home robot and an industrial robot is the adaptability to changes. While industrial setups are designed and optimized methodically, expected to stay unaltered, home environments change on a daily basis, introducing problems on-demand. This poses additional challenges in the implementation of robots in home environments: how should they be taught new tasks? Traditional program-testing-debugging cycle for robots is not applicable since it is too time-consuming and requires highly skilled users.

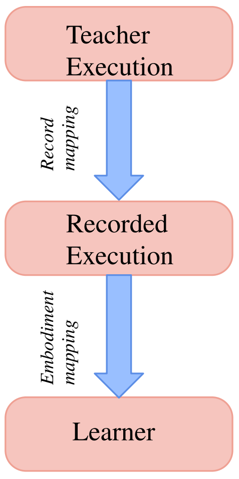
A popular way of dealing with this problem is Programming-by-Demonstration (PbD) [48]. In programming-by-demonstration, a human plays the role of a teacher and performs the task to be learnt by the robot. The robot observes the teacher and acquires a representation of the task, which can be used for later executions.

Formally, the goal of programming-by-demonstration consists of creating policies $\pi : Z \rightarrow A$ which map an observed state Z to an action A. This problem is solved as an instance of supervised learning, where pairs of states and actions (*demonstrations*) are used to learn the policy. For example, let us consider the problem of a robotic car learning how to take a curve at high speed. The observed state Z would be the current speed, curvature of the road ahead and angle of the wheels. The action A would be the angular velocity to apply to the steering wheel and the strength to apply to the brakes and accelerator. The goal of programming-by-demonstration in this case is to learn how to steer, accelerate and brake given the current state of the car.

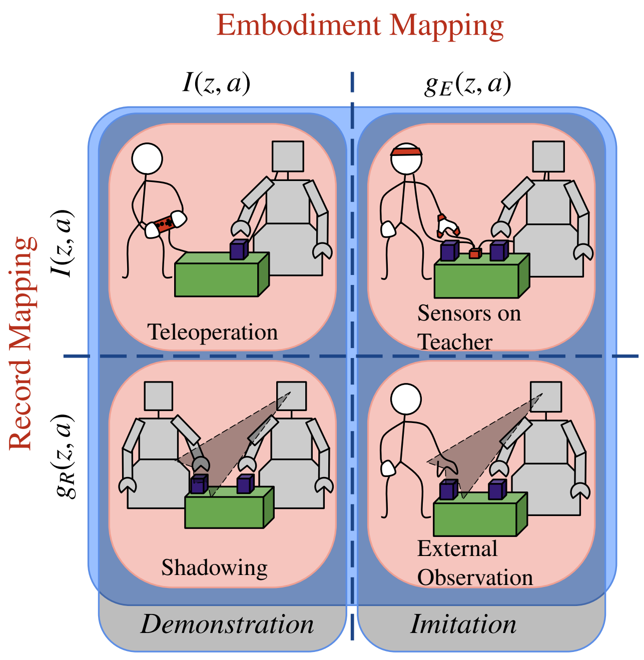
This represents the simplest case of learning-by-demonstration, because the pairs state-action acquired during the demonstration are directly usable by the learner: the states are directly observable and it is possible to execute exactly the same actions. The fact that the states and actions are provided to the learner exactly as they were observed by the teacher during the execution simplifies the learning process as well.

However, this is not always the case. Imagine that the learner robotic car has manual transmission instead of automatic. The actions of the teacher are not directly applicable by the learner. Similarly, if the teacher's car has a wheel locking sensor inexistent in the learner car, the states of the teacher have to be adapted before they can be used for learning. These differences in states/actions were described in [21] as non-trivial embodiment mappings $g_E(z,a)$.

Another situation which makes learning-by-demonstration more difficult is when the states/actions experienced by the teacher are not directly available. For example, since including sensors for the states and actions might be expensive, the learner can be given an aerial video of the demonstration instead. In this case, the actions and states have to be



(a) PbD Steps



(b) PbD Types

Figure 1.2: Programming-by-Demonstration classification according to [21]

inferred from the video. This inference procedure is defined as the record mapping $g_R(z,a)$.

In [21] different methods for gathering the state-action pairs are classified according to their embodiment and record mapping, see Figure 1.2. According to that classification, there are two main approaches: *Demonstration*, in which the states/actions recorded are exactly the ones the robot would observe/execute (as they are performed on the actual robot or an identical platform), and *Imitation*, where the states/actions are required to be mapped in order to be used by the robot. Each approach is further subdivided according to whether the demonstrations are directly ($I_R(z,a)$) or indirectly ($g_R(z,a)$) recorded from the teacher. We will describe in more detail the two extreme cases: teleoperation and external observation.

In *Teleoperation*, the robot is manually controlled by a human and it records his own sensory (states) and motor (actions) data. In this way, the pairs state-action are directly usable for learning, because the demonstrations were recorded on the learner's embodiment. Therefore, the record and embodiment mapping are the identity. One way of controlling the robot is through the usage of remote interfaces such as joysticks [164, 194]. Joysticks are mostly used for navigation purposes, where controlling the robots consists of telling them which direction to move to. In tasks such as manipulation, the large amount of variables to adjust through the remote interfaces makes it very difficult for the human teacher to efficiently execute the task. In these cases it is easier to interact physically with the robot, adjusting manually the position of the end-effectors (see Figure 1.3). This is called kinesthetic learning, and it is widely used in humanoid robotics.

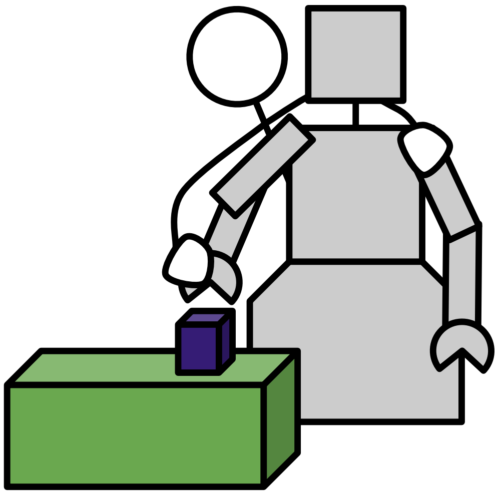


Figure 1.3: Kinesthetic Learning

Any embodiment or record mapping introduces imprecisions in the imitation systems. Since both the embodiment and record mappings are the identity ($I(z,a)$) in teleoperation, the demonstrated state-action pairs are accurately represented. However, since the teacher's knowledge about the robot's kinematics is limited, their execution of the task might not exploit appropriately the robot capabilities. The most common teleoperation method for complex robots, kinesthetic learning, requires passive movement and recording of the robot joint angles, which is not available in most of the robot arms.

At the other extreme we find *external observation*. In this paradigm, the learner does not have direct access to the teacher's states and actions, and therefore a non-trivial mapping $g_R(z,a)$ is required. Once the teacher's states and actions are inferred ("Recorded Execution" in Figure 1.2a), they need to be mapped to the robot embodiment through a non-trivial mapping $g_E(z,a)$. Both mappings potentially introduce inaccuracies in the final result. However, this method has three main advantages. First, it is less intrusive; the lack of sensors on the teacher makes the demonstration less constrained and faster to record. It also makes the setup of the "learning scene" faster. Second, it makes possible learning from already recorded material. For example, if a robot wants to learn how to open a door, and it only requires visual demonstration of the task, it can found many examples in youtube. Third, the hardware required for this approach is cheaper.

The remaining two paradigms consist of a mixture of direct and non-direct mappings. In *Shadowing*, the observations are mapped into the learner motor space, while the embodiment mapping is direct. An example of this methodology is [69], in which a robot learns how to navigate a maze by observing an identical teaching robot executing the task. *Sensors on Teacher* offers directly the motor commands of the teacher, but the recorded execution has to be non-trivially mapped to the robot's embodiment since it is not identical to the teacher's one [49, 112, 122, 138, 208].

1.1 Robot Grasping

In this thesis we consider the challenges involved in learning by demonstration for robot grasping. A substantial amount of work in robotics have been devoted to compute optimal grasps based on the geometry of the object. A common aim in robot control of grasping is to achieve a force closure grasp. A grasp is a force closure grasp if an external force is necessary to break contact between the object and fingers; any object motion or extra pressure applied by the fingers do not break such contact. Nguyen et al. [150] was one of the first to describe how to construct force closure grasps. Ponce et al. [163] extended this work to three fingers grasps, and sped up the algorithm to find object regions that ensure force closure grasps. Ferrari et al. [85] proposed two grasping quality criteria which maximizes the wrench that a grasp configuration can resist with minimum force applied with the fingers. Morales moved from predefined objects models to visually acquired models in [141], and from 2D to 3D models in [188].

All the previously mentioned approaches rely entirely on the perceived geometry of the object to be grasped. However, it is not always possible to infer the most adequate grasp solely based on geometry. Some aspects relevant for the grasp, such as temperature of the object (e.g. when grasping a frying pan), mass distribution (e.g. a hammer) or later usage (e.g. a jar from which liquid should be poured) cannot be extracted from the object's geometry. In connection to the last point, Borghi [42] showed that appropriate usage of objects is not necessarily related just to the object's shape. Moreover, Balasubramanian et al. showed in [32] that current grasping quality criteria performed worse in terms of grasping stability compared to grasps suggested by human subjects.

This evidence makes learning by demonstration techniques specially attractive for the task of robotic grasping. This approach, which we refer to as *Grasping by Demonstration* relates to the work of [189] that classify objects based on their affordances (categories like "sidewall-graspable"); through demonstrations, we are showing the learner what different objects *afford* in terms of grasping actions. We envision a scenario in which a robot learns which hand poses an object *affords* by observing a human grasping it. Other supervised learning methods can be used to "teach" robots how to grasp, such as providing a training set of objects with marked grasping points, and letting the robot to generalize this training set, [177, 79]. However, this method is less natural for the demonstrator than actually executing the grasps in front of the robots.

Grasping actions are usually part of more complex manipulation tasks, which themselves form part of goal directed activities. In this thesis, we concentrate on imitating the shape of the teacher's hand for a single grasping action. By doing this, we explore problems such as markerless pose estimation, correspondence between different embodiments and modeling of complex actions. Research in manipulation-from-demonstration focuses on other aspects, such as understanding *what* to imitate from a demonstration, [50], or how to generalize manipulation tasks for new objects, [161]. While one of our main research questions is how to shape the robotic hands while grasping, manipulation research usually treat hands as rigid objects.

Another topic, outside the scope of this thesis, is how to learn and imitate series of actions instead of single actions (e.g. [124]). One of the main problems in such scenario is how to segment the demonstrations into meaningful primitives; algorithms like Hidden Markov Models, [124], and nonlinear mixture of experts [215, 157] are two ways of extracting those primitives in an unsupervised manner.

There is a number of systems which have tackled grasping by demonstration in the past, [119, 162, 78, 18]. The system described in [162] designs a set of control laws whose parameters, i.e. the neutral pose of the hand and the its joint limits, are set by demonstrations. In [18], the classical grasp quality criteria, [85], is augmented with a task-related quality measure. A particular grasp is good in terms of this task-related quality criteria if the applied forces are similar to the forces applied in previous demonstrations. This grasp quality criteria requires the estimation of contact points between hand and object, which is particularly challenging in real scenarios where objects do not have tactile sensing surfaces. To solve this, a CyberTouch glove (a magnetic tracking device with touch feedback) is used to manipulate objects in a simulated environment, where it is significantly simpler to estimate contact points. Kang et al. [119] focuses on solving the embodiment mapping problem: he designed an algorithm which transfers human hand poses to robotic manipulators with different number of fingers. It senses the human hand pose with a Polhemus magnetic tracker, the hand configuration with a CyberGlove and the object position with a multi-camera system. This information is used to map the human hand configuration to a functional description of the grasp, applicable to manipulators with an arbitrary number of fingers. This description groups fingers into *Virtual Fingers*, each of which has a unique role in the grasp. A similar approach is followed in [78] where the hand configuration is observed through a CyberGlove and mapped into different robotic hands (Barrett and Robonaut) by first labeling the grasp as one from a discrete set and then mapping the configuration based on the grasping class.

All the systems described above have in common a trivial record mapping $I(z, a)$ since the sensors are placed on the teacher. While [162] follows the *Teleoperation* paradigm as the controlled manipulator is has the same embodiment as the teacher, the rest are instances of *Sensors on Teacher*, where a non-trivial embodiment mapping $g_E(z, a)$ has to be used. As discussed before, the advantages of trivial record mappings come at the cost of specialized hardware and unnatural demonstrations by the users. For these reasons we target a system following the *External Observation* paradigm.

1.2 Contributions and outline

Our vision of a grasping by demonstration system based on the *External Observation* paradigm is shown in Figure 1.1. The record and embodiment mapping described in [21] are implemented by the “Hand Pose Estimation“ and “Human2Robot Mapping” respectively. The temporal scheme proposed in [21] (Figure 1.2a) is extended with an extra component: an execution part which ensures the grasping is performed as planned through a closed perception loop. A fourth component that embraces the three previous ones is the modelling of the grasping actions. Modeling grasps can help sensing them accurately, mapping them to different embodiments and finally executing them successfully.

Each of the modules shown in 1.2a corresponds to different active fields of research in Computer Vision and Robotics. This thesis contributes to each of those fields, tackling specific aspects of the problem that are of particular importance for Grasping by Demonstration.

This thesis is organized following the block division depicted in 1.2a: Sensing Human Grasps, Grasp Modeling, Human-to-Robot Mapping and Grasp Execution.

- **Chapter 2: Sensing human grasps**
  This chapter describes the first building block in our system. It implements the record mapping $g_R(z,a)$, translating sensory data into human motor representation. This problem has been studied intensively in computer vision [100, 43, 26, 173, 66, 68, 190, 67]. However, the majority of those systems focus on estimating the pose of a free hand without computational restrictions. Some robotic solutions to the hand pose estimation problem like [171, 151] provide real-time estimations, but generally introduce additional restrictions.

  We describe two approaches to the hand pose estimation problem. First, in 2.3 we describe how to reconstruct 3D hand contours as a building block for a system similar to [151]. Despite of improving the performance of state-of-the-art systems such as [22], the lack of robustness with respect to occlusions made us abandon this path. Second, we present a discriminative approach which successfully estimates a grasping hand pose in real time. The method compares the images from a monocular camera with a database populated by synthetically generated images of grasping hands. This represents to the best of our knowledge the first hand tracker robust to occlusions which runs in real-time.

- **Chapter 3: Modeling grasping actions**
  The articulation of human hands is usually model with 20-22 degrees of freedom. Such high dimensionality makes processing hand poses computationally expensive and prone to overfit (*curse of dimensionality*). However, the poses are correlated in time (a hand pose is not independent of previous poses) and its dimensions are correlated with each other (e.g. the position of thumb and index fingertip are not independent). The estimation of human hand pose can provide us insights about these correlations, which can be used for improving different aspects of our grasping by demonstration system. For example, temporal correlations can be used to improve our hand pose estimation system by rejecting poses which are highly unlikely according to the model. Correlation between fingers can be exploited in areas such as robotic hand design, permitting the control of multiple fingers with a small amount of signals.

  In this chapter we aim at extracting evidence from real human data in order to acquire models of human grasps. More specifically, we exploit the concept of *postural* *synergies* for grasping: low dimensional manifolds that describe grasping actions. Differently from previous studies, we examine time-varying hand poses (instead of modeling only the final grasping pose, [175]) from multiple grasps performed by multiple subjects. These grasps are represented in a common space, as opposed to the usual approach of modeling every grasp and/or subject individually, [98, 136].

The main focus of this chapter is on the assumptions that different postural synergy representations make on data and how they affect the quality of the extracted models. We transcend the classical methodology for extracting postural synergies and consider the usage of modern generative methods such as Gaussian Process Latent Variable Models (GP-LVM). We evaluate our models quantitative and qualitatively, and show how they can be used for evaluating the quality of hand kinematic designs.

- **Chapter 4: Human-to-Robot Mapping**
  A major challenge that our approach faces is the non-trivial mapping from human motor representation to robot motor representation $g_E(z, a)$. This mapping relates to the "Correspondence Problem" [148, 147]: how imitation can occur when the learner embodiment is not the same as the teacher's one. This topic has been covered extensively in biology [102, 158], but has not receive so much attention in robotics. As stated in [148], the central component of imitation is a correspondence measure that tells us how successful an imitation strategy has been.

  In this chapter we describe a correspondence measure for grasps performed with different hands, based on the concept of *Virtual Fingers* [20]. We also provide a mapping of human grasping actions to a robot. Further, we evaluate the mapping based on the correspondence measure described before. Finally, we show how such mapping can be applied to grasping for a standalone robotic arm and with a humanoid robot.

- **Chapter 5: Grasp Execution**
  The execution of robotic tasks commonly follows the perceive→plan→act paradigm. However, the environment in which we envision robots acting is highly dynamic, meaning that the perception acquired before planning might not be valid in the action stage. Moreover, this scheme is very sensitive to inaccuracies in the perception and planning stage, which are prone to happen. We believe that continuous perception during the planning and execution of robot tasks makes the system more robust with respect to inaccuracies and dynamic changes.

  In specific, the mapping described in Chapter 4 is based on an initial estimation of the object position with respect to the robot. However, the correspondence measure clearly depends on details which can only be accessed during the execution of the task, such as the position of the fingertips. Continuous perception of those variables could be used to correct our initial plans.

  In this chapter we describe two methods that aim towards closing the prediction-perception-action loop. Firstly, we show how simple corrective movements improve the performance of the mappings described in Chapter 4. Secondly, we show how to correct robot movements based on the mismatch between real images of the robot and our prediction of its appearance based on forward kinematics. This corrections allows us to overcome calibration errors that are inevitable in our robotic setup.

- **Chapter 6: Summary and Discussion**
  This chapter revisits the contributions and future lines of work elicited by each of the modules described in this thesis. A compact view of the techniques used in our work is provided. Finally, we conclude with a our thoughts about the remaining challenges concerning grasping by demonstration.

1.3 Publications

Parts of this thesis have previously been published in the following journal and conference articles.

[1] **J. Romero**, D. Kragic, V. Kyrki, and A. Argyros. Dynamic time warping for binocular hand tracking and reconstruction. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 2289 –2294, may 2008.
The author of this thesis extended and improved the system from A. Argyros for 3D contour reconstruction.

[2] **J. Romero**, H. Kjellström, and D. Kragic. Modeling and evaluation of human-to-robot mapping of grasps. *International Conference on Advanced Robotics (ICAR)*, 2009.
This article was based on work from the author of this thesis.

[3] **J. Romero**, H. Kjellström, and D. Kragic. Monocular real-time 3D articulated hand pose estimation. In *IEEE-RAS International Conference on Humanoid Robots*, 2009.
This article was based on work from the author of this thesis.

[4] **J. Romero**, H. Kjellström, and D. Kragic. Hands in action: real-time 3D reconstruction of hands in interaction with objects. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 458–463. IEEE, 2010.
This article was based on work from the author of this thesis.

[5] **J. Romero**, T. Feix, H. Kjellström, and D. Kragic. Spatio-temporal modeling of grasping actions. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS*. IEEE, 2010.
This article was based on work from the author of this thesis in collaboration with T. Feix.

[6] **J. Romero**, T. Feix, C. H. Ek, H. Kjellström, and D. Kragic. Extracting postural synergies for grasping. *Submitted to IEEE RAS Transactions on Robotics*, 2011.
This article was based on work from the author of this thesis in collaboration with T. Feix.

[7] H. Kjellström, **J. Romero**, D. Martinez, and D. Kragic. Simultaneous visual recognition of manipulation actions and manipulated objects. *European Conference on Computer Vision (ECCV)*, 2008.
The author of this thesis helped with the experimental evaluation of the method.

[8] H. Kjellström, **J. Romero**, and D. Kragic. Visual recognition of grasps for human-to-robot mapping. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS*, 2008.
The author of this thesis was in charge of the experiments related with the robot.

[9] M. Do, **J. Romero**, H. Kjellström, P. Azad, T. Asfour, D. Kragic, and R. Dillmann. Grasp recognition and mapping on humanoid robots. In *IEEE-RAS International Conference on Humanoid Robots*, 2009.
The author of this thesis provided the grasp classification framework in this system, as well as its integration, as well as writing the final version of the article.

[10] H. Kjellström, **J. Romero**, and D. Kragic. Visual Object-Action Recognition: Inferring Object Affordances from Human Demonstration. *Computer Vision and Image Understanding*, 2010.
The author of this thesis helped with the experimental evaluation of the method, and provided input in terms of hand pose estimation.

[11] T. Feix, R. Pawlik, H. Schmiedmayer, **J. Romero**, and D. Kragic. A comprehensive grasp taxonomy. In *Robotics, Science and Systems: Workshop on Understanding the Human Hand for Advancing Robotic Manipulation*, June 2009.
The author of this thesis designed the rendered poses for the grasps.

[12] X. Gratal, **J. Romero**, and D. Kragic. Virtual visual servoing for real-time robot pose estimation. In *18th IFAC World Congress*, 2011.
The author of this thesis worked on the visual features in the system and on its general refinement.

[13] X. Gratal, **J. Romero**, J. Bohg, and D. Kragic. Visual servoing on unknown objects. In *IFAC Mechatronics: The Science of Intelligent Machines*, 2011.
The author of this thesis worked on the visual features in the Virtual Visual Servoing system and on the system's general refinement.

[14] T. Feix, **J. Romero**, C. H. Ek, H. Schmiedmayer, and D. Kragic. Visualization of anthropomorphic hand performance. *Submitted to IEEE RAS Transactions on Robotics*, 2011.
The author of this thesis collaborated in terms of the underlying dimensionality reduction system, as well as ideas and writing the final version of the article.

Chapter 2

Hand Pose Estimation

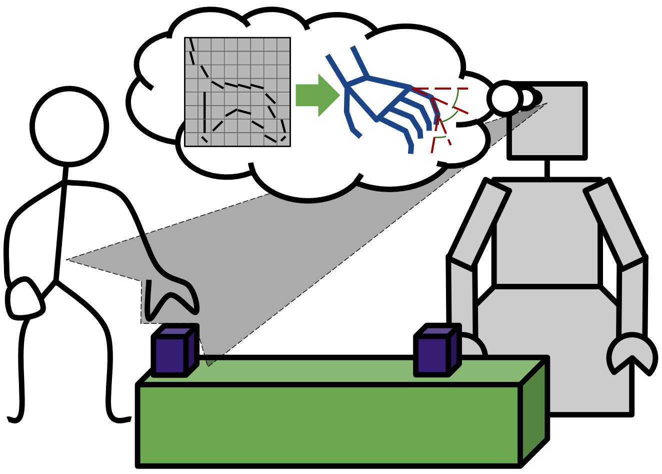


Figure 2.1: Robot estimating human hand pose. The robot observes through its camera(s) the demonstrator's hand, extracts features and infers the pose of the human hand as joint angles of its internal human hand model

It is a common practice in robotics to show the difficulty of a particular task by describing cases which are not easily solvable even for humans. For example, we will use such an argument in order to show the difficulty of acting in the absence of sensory feedback in Chapter 5. It is hard though to apply that argument for hand pose estimation, given the degree of expertise that humans exhibit when imitating hand poses based on visual input. However, this does not mean that automated hand pose estimation is an easy problem. First, we should consider that imitation capabilities in humans involve both pose estimation and refinement of the pose; inaccuracies in the estimation are alleviated by posterior closed-loop optimization of the pose (more about that in Chapter 5). Secondly, automated hand pose estimation is as difficult as, if not more difficult than body pose estimation, since it has comparable dimensionality and a higher degree of self-occlusion. Body pose estimation is widely recognized as very demanding, and is actively studied [31, 184],

We should interpret its apparent ease for humans as evidence for the evolutionary importance of hand pose estimation. Given that learning by imitation in monkeys is a successful technique for acquiring new behavior [45], we can argue that imitation (and consequently hand pose estimation) has probably played an important role in the evolutionary development of tool usage in humans.

Our goal in this chapter is to develop methods that enable the robot to perceive human hand poses, which is of key importance for grasping-by-demonstration.

The usage of devices such as visual markers [196, 130] or gloves [195, 18] is a popular way of simplifying the estimation of human hand poses. After the position (and sometimes the orientation) of these markers are obtained, the skeleton of the teacher can be computed through inverse kinematics. Simpler devices like textured gloves [214] also achieve good results. These methods deliver very accurate results, but their setup is time-consuming and constraining, potentially interferring with the task performance. Moreover, the usage of these kind of devices makes it impossible to gather certain types of data: for example, it is not possible to recollect data about dish washing with a magnetic glove.

Another possibility is to infer the teacher's body configuration directly from images or depth-maps, without placing any markers. Although this approach is clearly more challenging, it overcomes the disadvantages of the marker-based approach previously mentioned, at the cost of a potential lower accuracy. Another advantage of the markerless approach is the availability of abundant markerless footage of executions of (virtually) any kind of task, allowing a robot to learn from multiple sources just by surfing the web. This is the approach that we want to pursue in this thesis for the purpose of human hand pose estimation.

2.1 Challenges in markerless hand pose estimation

3D reconstruction of human motion from images and video is a non-trivial problem, characterized by high-dimensional state spaces, fast and non-linear motion, and highly flexible model structures [140]. The implementation of a markerless hand pose estimation system presents a number of challenges: the high dimensionality of the human hand pose space, the high degree of occlusions (both self occlusions and object occlusions), and the velocity of human hand movements.

The human hand can be considered as a 20-22 dimensional articulated body [172, 44, 72]. Such a high dimensionality poses a problem for pose estimation systems, since the number of possible hand poses is huge. This dimensionality has been reduced in some systems [129, 172] after observing the coupling of some finger joints. However, these approximations can be too rough for intricate control of the hand [80]. Another approach for reducing the dimensionality of the hand is to model the hand in a manifold of its pose space, previously extracted from training data [175, 56, 207]. These manifolds are trained with poses corresponding to a particular task, in this case grasping. They can be represented non-parametrically by their samples [26, 173, 4]. This circumvents the problem of dimensionality by sampling only relevant examples, which lie in a lower dimensional space. However, the accuracy is then limited by the granularity of the sample set.

One of the main differences between full-body pose estimation and hand pose estimation are the occlusions, both self and externally inflicted. While full-body pose estimation system deal with mostly covered bodies that make the foreground-background segmentation challenging, hand pose estimation has to deal with more severe object and self occlusions. Despite of the importance of object occlusions, researchers have almost exclusively focused on estimating the pose of hands in isolation from the surrounding scene, e.g. [66, 200, 201, 214, 26, 190]. On the other hand, self occlusions make the pose estimation problem ill-posed, since many different poses have similar appearance when potentially discriminating parts of the structure are occluded. Disambiguating between them usually relies on temporal filters [190]. While object occlusions introduce ambiguity as well in the sense that the occluded part of the hand could be in any state, they pose a different problem. Object appearance is much more variant than hand appearance, which represents a problem for systems modeling explicitly the appearance of the hands. A solution to this is to use local trackers that can overcome the loss of track of some hand parts [100]. Our solution proposed in [5] consists of including typical object occlusions in the non-parametric model of the hand; these typical occlusions can not only account for the missing parts of the observed hand, but to improve the pose estimation performance since specific occlusions are observed typically in connection to specific poses.

The third challenge in human hand pose estimation corresponds to the dynamical behavior of human hands. State of the art in hand pose estimation have rather simple dynamical models for the hand, such as zero velocity ([190] for global motion, [192]), static velocity ([100] for the finger motion, [67]) or Markov-1 ([190] for the finger motion). Moreover, the speed of the hand (5 m/s translational and 300 °/s rotational speed of the wrist [81]) often violates the common assumption of small change from frame to frame. Simplicity in the models is generally preferred since more complex models can be restrictive [190]. Another dimension of the motion modelling problem to be considered is the selection of the space where the motion should be parametrized. There are mainly two approaches: modeling the motion of parts of the hand (task-space, [192, 100]) and modeling the motion of joints of the hand (joint-space, [190, 67]). On one hand, modeling the motion of joints angles represents a lower dimensional problem, and enforces that the parts are configured according to the kinematic chain. On the other, the motion of hand parts easily enforces that fingers should not collide with each other. Moreover, measuring similarity in joint angle space can result in undesired results, since some joints have a larger impact in the hand appearance than other ones, [74].

Finally, a remaining challenge for hand pose estimation is to perform the estimation in real time. Although some tasks such as video analysis do not depend on real time performance, such a characteristic is a requirement for interactive grasping by demonstration sessions.

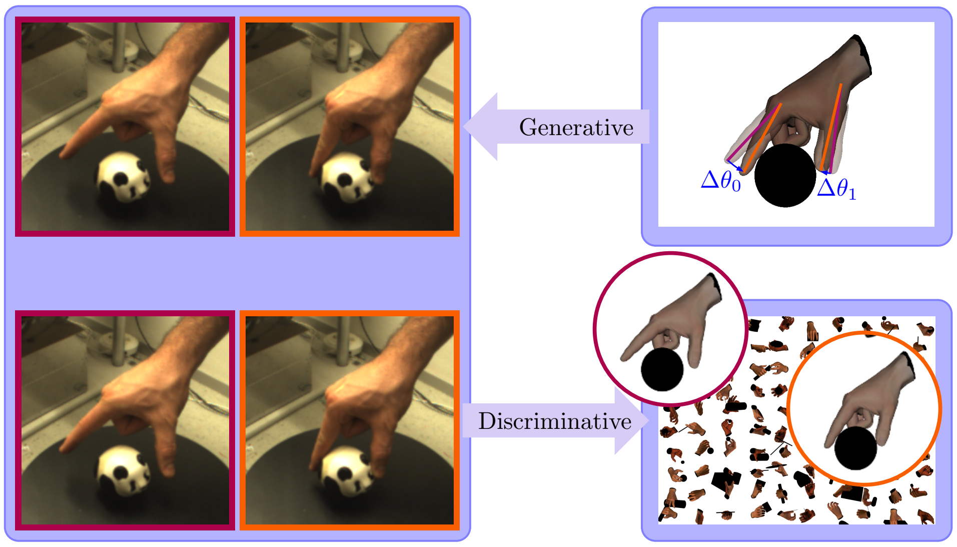


Figure 2.2: Given the real images on the left, generative models (top) try to model a representation which can generate those images. In this case, given a synthetic model of the hand, this consists in estimating the θ_i angles (or the variations $\Delta\theta_i$). Discriminative pose estimation (bottom) does not require an underlying model capable of generating new poses; the poses can be obtained from a set of possible poses, for example.

2.2 Related work

It is very hard to solve all the challenges stated previously. Therefore, the existing hand pose estimation systems only address one or a few difficulties. This can be done by posing different restrictions on the tracker, such as only allowing a limited range of motions. In the following paragraphs we will give an overview on what has been done in the field of hand pose estimation. We can roughly divide the hand pose estimation systems into two groups: generative systems and discriminative systems.

2.2.1 Generative systems

Generative systems (Figure 2.2) rely on the existence (and online usage) of a model that can *generate* the appearance of the hand and compare it with the perceived hand. In mathematical terms, a model that transforms a human motor representation $\hat{X}$ into a perceptual representation $\hat{Y}$ (typically visual features) is used online to minimize the perceptual error $||\hat{Y} - Y||$ and thus in this way minimize the motor error $||\hat{X} - X||$. The difficulty of the error minimization $||\hat{Y} - Y||$ depends heavily on the perceptual representation Y used. In this respect we contemplate two main approaches: high level features (such as fingertip positions), or low level features (such as edges, image moments, etc). On the one hand, low level features are easy to extract and robust to occlusions and other non-ideal conditions; however, the minimization of the perceptual error is not trivial. On the other hand, high level features are difficult to extract, but they substantially simplify the error minimization problem, providing sometimes a closed-form solution for it.

A system using high level features is described in [151]. This system detects the fingertip positions in 2D and, based on those positions, computes the motor parameters of the human hand as joint angles. The detection of the fingertips is done in a hierarchical fashion: after reducing the dimensionality of the image through the usage of sparse gabor filters, one layer of a neural net locates roughly the fingertip positions; subsequent layers refine the location of the fingertips. A parametrized self-organizing map is used to invert the forward kinematics from the articulated model of the hand. The resulting inverse kinematics compute the human hand joint angles from the fingertip positions computed in the previous step. The extraction of the motor representation $\hat{X}$ from the perception $\hat{Y}$ is rather simple; even a closed-form solution could have been used for the particular hand joints constraints used in this system. The extraction of the fingertip positions is where the main research of this system is, and where most of its limitation are. The system makes the implicit assumption that all fingertips are visible all the time, which does not hold for some hand configurations, and fails often in the scenario of object manipulation. Moreover, it is unknown how would the system perform when the hand orientation is not fixed by the hardware setup. While the system presented in Section 2.3 has similar limitations, in Section 2.4 we show how the usage of low level features overcomes them.

Some systems like [183, 171] try to overcome the lack of robustness of high level features by correcting their estimation $\hat{X}$ based on the appearance error $\hat{Y} - Y$. These systems also depend less on the fingertip positions by taking into account the silhouette of the hand as well.

However, since the extraction of fingertip positions is unreliable in realistic setups, most of the systems extract only low level features which are present no matter which viewpoint or which part of the hand is occluded. These systems extract features such as silhouettes [125], orientation gradients [88, 223], edges and pixel colors [222, 193, 190], edges and motion flow [134], pixel colors [67, 205] or pixel depths [100, 43]. We will examine in detail three recent state-of-the-art systems.

Pixel colors can be used directly to compute the estimation error, as it is described in [67]. Such a system uses a detailed model of the human hand to estimate the hand pose and texture, as well as the lighting configuration of the scene. This estimation is done by calculating analytically the gradients of the image error (which is the pixel color error) with respect to the variables to be estimated, and performing a gradient descent optimization. The discontinuities in the pixel colors, due to occlusions and self-occlusion, require a special treatment in this approach. Therefore, the inclusion of objects in the scene that would occlude the hand might require an accurate model of the object, as well as an accurate estimation (within the same framework) of the object pose. Another disadvantage of this approach is its low speed, since it requires the calculation of the error gradient with respect to a large number of parameters.

As mentioned in [67], one of the key problems in monocular hand tracking concerns the existence of depth ambiguities. In that system this is solved by using illumination and shading information. However, depth information can be directly extracted from devices like time-of-flight sensors [15] or structured light sensors [82, 139]. In the last years some systems have used pixel depth information directly from such sensors to overcome the depth ambiguities. The system described in [43] minimizes the distance from a subset of points on the hand estimated mesh model to the real depth image. The minimization is perform by Stochastic Meta-Descent, a gradient descent method which estimates the gradient from a stochastic set of points and optimizes the hand parameters and the step size. A set of those trackers are treated as particles in a particle filter scheme to track simultaneously multiple hypothesis.

A time-of-flight sensor is used in [100]. This approach is radically different to the previous ones: in order to improve robustness with respect to occlusions, the tracking problem is dividing into local trackers that are then joined by belief propagation. The hand model is treated as a set of rigid bodies (three bodies per finger), each of them associated to an individual tracker. Each of these trackers estimates the likelihood of the observation given the 6D (position and orientation) pose of the body in a set of uniformly sampled poses. This likelihood is an exponential function of the 3D error between the observed 2.5D image and the rendered image of the finger link. Those likelihoods are combined in a belief propagation fashion in order to take into account the kinematic (human joints are limited to a certain range) and proximity (adjacent links should be close to each other) constraints. Collision between links are not considered in this scheme since it would make the belief propagation slower, and it was not necessary according to the authors. Since the complexity of belief propagation scales quadratically with the number of samples in each tracker, this was reduced by selecting the most likely subset of samples and iterating the process.

The three systems described before have in common that the process of the low-level features is too slow to make the system work at camera frame rate. Two of them [67, 43] rely on gradient descent optimization which need expensive calculations of the gradient. The third [100] needs computing the likelihood of multiple poses for each of the finger links and combine this information with a (computationally expensive) belief propagation scheme. Moreover, all the systems require initialization since they are inherently local methods. Stenger et al. concludes in [190] that “the time required for projecting the model is approximately three orders of magnitude higher than evaluating the likelihood function”. This is quite general for generative systems that compare visually the real observations with their estimations. That makes particularly attractive the paradigm of discriminative methods: generate samples for the model appearance offline.

2.2.2 Discriminative systems

Discriminative (Figure 2.2) systems model the appearance of the hand as a discrete collections of samples, instead of having a continuous generative model capable of generating the hand appearance for any pose. In this way they save computational time since generating the hand model appearance is computationally expensive, at the cost of potentially losing accuracy.

These systems are the natural choice for classification tasks such as sign language or finger spelling recognition [84, 205]. For tasks like full pose recognition, interpolation can be used to provide continuous pose estimation [173]. The databases used for classification can be composed by real images [216] or synthetic images [173, 190]. Although synthetic images are less similar to the test data than real images, large synthetic sets can be constructed much easier than large real sets, and they provide pose ground-truth data. In [216] the small size of the real dataset is alleviated by the use of a mixed supervised-unsupervised learning, which enlarges the dataset substantially with real unlabeled data.

One of the first well-known discriminative hand pose estimation systems is depicted in [173]. Their dataset is composed by synthetic images depicting poses extracted from humans with a CyberGlove and rendered from multiple view points uniformly distributed on a sphere. They developed a technique called Specialized Mapping Architectures (SMA) that, given a dataset with hand poses and visual features, computes a set of functions that locally map the hand visual features (Hu moments [108]) to the hand pose parameters (22 joint angles). The actual function to be used for a specific input is chosen by computing the corresponding visual feature for each mapping and choosing the one which minimizes the visual error. Since the functions output is not restricted to the training dataset, it can be considered that the system performs regression more than classification.

A more recent discriminative system is described in [190]. The system computes the maximum-a-posteriori (MAP) estimation of the hand pose each frame based on a set of synthetically generated hand templates. The probability of each template is decomposed into an observation likelihood and a transition likelihood. Since the computation of these likelihoods for all the templates would be prohibitively expensive, the estimation is arranged in a hierarchical fashion. The pose space is divided into a hierarchy of regions; at the first level, the likelihood of the center of each region is evaluated, and the descendants of regions with likelihoods below a certain threshold are discarded from farther evaluation. Since the pose space is discretized, the transition likelihood can be stated as a Markov state transition matrix, estimated from human training data. The system estimated up to 8 DOF with databased up to 35000 templates in the leaf level, reporting speed-up with respect of the exhaustive search of two or three orders of magnitude.

2.3 Obtaining 3D contours for high level feature extraction

As previously mentioned, extracting high-level features can substantially simplify the estimation of hand poses. The usage of a single camera in [151] comes at the cost of strong restrictions on the hand pose. Without those restrictions, the systems might not be able to perform well given the depth ambiguities inherently present in monocular camera setups. In [1], we envision a system that extracts 3D silhouettes of the human hand as a first step towards extracting higher level features for hand pose estimation. The systems uses a robotic head with a stereo camera system on it (see Figure 2.3) to compute the 3D hand contours. Once hand contours are identified in each of the stereo images, the points along the contours have to be matched in order to compute their 3D position. The goal of [1] is to show how matching and reconstruction of contour points using Dynamic Time Warping (DTW) outperforms algorithms such as Iterative Closest Point (ICP). The remaining of this section describes the contributions of [1].

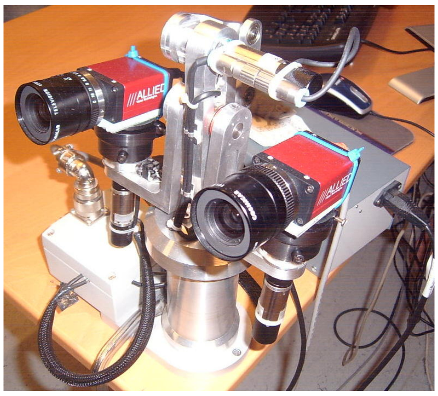


Figure 2.3: Stereo head used in our system

2.3.1 Introduction

Depth information is important for generative methods. Although the first hand pose estimation systems relied on single camera systems ([17, 171, 217, 151]), the availability of depth information from either multiple camera systems ([66, 68]), time-of-flight sensors ([100]) or structured light ([43]) can constrain further the problem of hand pose estimation. Although time-of-flight and structured light present advantages over stereo-based system in non-textured surfaces, the 3D reconstruction of such system in the object edges is noisier that the one obtained from stereo. Moreover, 3D reconstruction systems were considerably more expensive than stereo systems at the time when this system was designed, before the arrival of Kinect sensors [139]. Another advantage of multiple cameras setups is their robustness to occlusions.Stereo "humanoid" heads like the one we use, see Figure 2.3, can effectively avoid some of the occlusions. If the camera baseline is large and/or the number of cameras is more than two as in [66], even some significant occlusions can be tolerated. This is important in hand tracking since the self-occlusion is a common problem.

This work is based on further developments of a system presented in [23]. In that system, the hands are first identified separately in each of the stereo images and their contours are extracted. This is followed by stereo-based blob matching techniques and shape matching through contour alignment. The particular objective of the work presented in this section is the development and evaluation of the shape matching and contour alignment step. In [23], the two contours coming from the stereo cameras are matched with ICP algorithm, under the assumption of affine transformation (translation, shear and scaling). However, this approach is not suitable for cases where the inherent assumption of planarity of the hand due to the affine motion model is not valid. This is commonly the case in object grasping and manipulation activities or when a full 3D pose estimation of the hand is required. This section presents a new approach for contour alignment based on dynamic programming, considered in this section in the DTW context [146]. The approach is not dependent on the validity of the planarity assumption and an extensive experimental evaluation shows that the performance of the new approach is clearly superior, increasing the robustness to occlusions and relaxing the planarity assumptions. While dynamic programming approaches have been widely used in dense stereo matching, we are not aware of applications in closed contour based matching in stereo.

We will now introduce the contour matching algorithms compared in this Section, namely ICP and DTW.

2.3.2 Iterative Closest Point (ICP)

The ICP algorithm computes a motion which transforms one set of points into another one according to a model. The algorithm is iterative: At each iteration, it first computes a motion which minimizes the current matching error and then applies the estimated motion to update the point correspondences. Two strong assumptions are made:

- An initial estimation of the point correspondences is available.
- A suitable motion model is available to perform the contour alignment.

In [23], the initial error measure is based on the first and second moments of the contours. The hand contours are approximated by ellipses and their centroids and principal axes are matched. This approximation is fast, but it has some problems: the more circular this ellipse approximation of the contour is, the more uncertain is the alignment of the axes. Affine transformations are used to model the motion in [23]. These transformations are represented by six parameters, and can model properly the transformation of planar shapes between stereo-views. This restricts the hand contours to be extracted substantially. The problem can be solved by choosing a more general transformation, although more general transformations require more parameters to be optimized. For this reason, the number of iterations until the convergence is achieved may increase drastically. This algorithm is relatively fast, since it computes a motion that is applied to all the points at the same time. On the other hand, it is iterative and sometimes the number of iterations required to reach the convergence may be high.

2.3.3 Dynamic Time Warping (DTW)

The approach adopted here, Dynamic Time Warping (DTW), is conceptually different. It is not iterative, and it computes the point correspondences without any assumption of the underlying motion model. The algorithm consists of four steps:

1. Compute the pairwise distances from each point in one set to all the points in the other set.
2. Select a pair of points that are supposed to be a good match, in order to initiate the matching process.
3. For each possible pair of points, compute the accumulated cost of reaching it, based on the accumulated cost of previous points and the cost of the jump from the previous pair (the pair with minimum accumulated cost previously computed).

4. The optimal path corresponding to the minimum total cost of matches can be extracted by tracing back from the end point.

The algorithm has time complexity $O(n^2)$ where n is the number of points to be matched. In the remaining part of the section, we present the principal details of the proposed approach: the building of the distance matrix and the choice of the starting point. More information about DTW can be found in [146].

2.3.3.1 Distance Matrix

DTW algorithm finds the set of correspondences with the least total cost (or distance) between the matched points. For this reason, it is very important to choose a distance measure that in a good way represents the similarity of two hand contours in a stereo pair of images.

From a geometric point of view, the relation between points in a calibrated stereo pair is given by the essential matrix. For a given point, this matrix provides the epipolar line on where the corresponding point must lie in the other image. This matrix depends on the extrinsic calibration of the cameras (the translation vector and the rotation matrix between them):

$$P_R^{\mathrm{T}} R[t]_x P_L = 0 \Leftrightarrow P_R^{\mathrm{T}} E P_L = 0 \tag{2.1}$$

$$[t]_x = \begin{pmatrix} 0 & -t_z & t_y \\ t_z & 0 & -t_x \\ -t_y & t_x & 0 \end{pmatrix} \tag{2.2}$$

The relation is applied to points P_R and P_L in the normalized camera coordinate systems, and are also valid for their normalized image plane projections p_R and p_L. We are interested in this relation expressed in the image coordinate system in order to determine the correspondence between the left and right camera images. Considering the camera intrinsic parameters K^{-1}, we obtain the well known fundamental geometry relationship:

$$\begin{aligned}
p_L = K_L^{-1}\overline{p}_L \qquad p_R &= K_R^{-1}\overline{p}_R && (2.3)\\
p_R^{\mathrm{T}} E p_L &= 0 && (2.4)\\
\overline{p}_R^{\mathrm{T}} K_R^{-\mathrm{T}} E K_L^{-1} \overline{p}_L &= 0 && (2.5)\\
\overline{p}_R^{\mathrm{T}} F \overline{p}_L &= 0 && (2.6)\\
\overline{u}_R &= F\overline{p}_L && (2.7)
\end{aligned}$$

The points $\overline{p}_R$ and $\overline{p}_L$ represent pixel coordinates and $\overline{u}_R$ represents the epipolar line where $\overline{p}_R$ must lie. The fundamental matrix F can be computed from the intrinsic and extrinsic parameters of the camera or estimated directly from a set of calibration images. The calibration of the stereo rig was performed with the tool available from [191].

The distance matrix is built from the pair-wise distances between points. Those distance are measured as the distance from the first point to the epipolar line generated by the second point. The main advantage of this measure is that it is a direct consequence of the stereo-set geometry, and guarantees that the distance between matching points is zero if the cameras are properly calibrated. The main disadvantage is that this measure might be ambiguous: the point denoted 0 in Figure 2.4, it will perfectly match both points denoted 0 and 8 in another image.

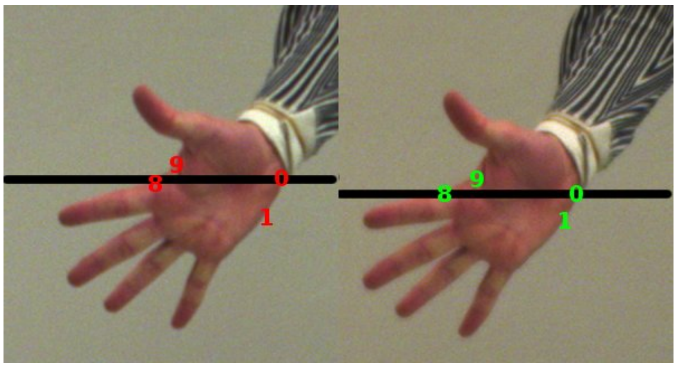


Figure 2.4: Point 8 has distance 0 to point 0 since it is based on the distance between the epipolar lines the points lie on.

However, this problem can be solved by the DTW algorithm. Although two points would have similar distances to each other, the subsequent ones will probably have very different distances (as points 1 and 9 in Figure 2.4). This means that the boundaries of such ambiguous cases can help solving them. It is only when the ambiguities extend for several points that the system can produce erroneous results. This can be seen in the third row of Figure 2.10, where the wrist segment of the contour is wrongly reconstructed since it is almost parallel to epipolar lines.

DTW is also robust to occlusions. If some points are occluded just in one of the images of the stereo pair, the number of contour points will be different in each image. When the algorithm reaches the point where the partial occlusion begins, it detects that the distance between the next pairs increases until the occlusion finishes. If a suitable distance measure is chosen, DTW will drop these points, that is, it will leave them without any correspondence in the other image. The matching will continue normally when contour points are visible in both images again, see Figure 2.5.

2.3.3.2 Starting Point Selection

Matching cyclic sets of points with DTW has one prerequisite: the beginning and end matching pair have to be chosen and it has to be the same point pair. Although DTW can align sequences with disaligned starting points, the points that are dropped while aligning the sets are lost. In Figure 2.6 we show an example of that: the real starting point pair should be the pair $(5,0)$ instead of $(0,0)$. As the consequence, only the central point pairs of the contours, from $(5,0)$ to $(14,9)$, are well matched. There are solutions to this problem,

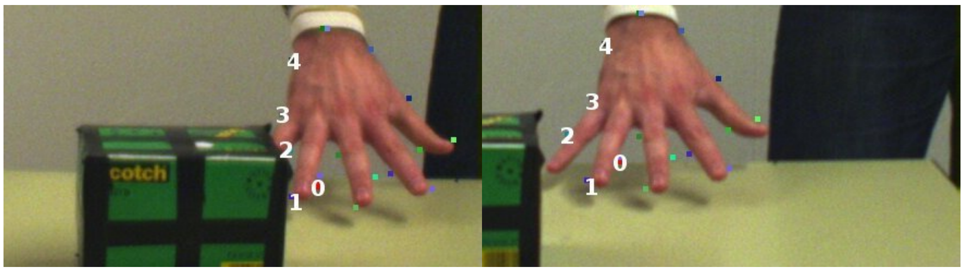
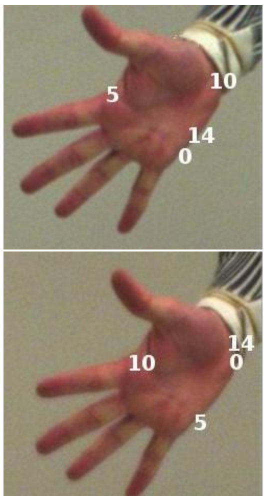


Figure 2.5: Only point 2 is wrongly matched due to the occlusion.

but they usually require at least twice the computation time required by the original DTW algorithm.



14	10	9	8	6	2	7	8	12	16	18	20	21	15	14	**10**
13	12	7	6	1	5	6	7	8	11	13	17	19	13	12	**14**
12	9	7	0	5	9	14	15	16	18	20	21	15	14	12	**11**
11	5	1	2	7	9	14	16	18	20	21	15	14	10	9	**9**
10	1	3	7	9	14	15	16	18	20	21	15	14	10	9	**8**
9	4	6	7	8	11	13	17	19	13	12	9	8	7	6	**2**
8	7	9	14	15	16	18	20	21	15	14	10	9	6	**1**	7
7	6	7	8	11	13	17	19	13	12	9	8	3	**0**	9	14
6	15	16	18	20	21	15	14	10	9	5	2	**1**	5	5	14
5	18	16	11	13	17	19	13	12	9	4	**1**	3	6	9	18
4	11	13	17	19	13	12	9	8	6	**2**	5	10	11	12	14
3	21	15	14	10	9	8	8	7	**0**	4	9	11	10	11	12
2	21	15	14	10	9	6	5	**1**	7	9	11	10	11	12	13
1	17	19	13	12	9	6	**1**	2	6	9	11	10	11	12	13
0	**15**	**12**	**10**	**6**	**3**	**0**	3	6	9	11	10	11	12	13	17
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

Figure 2.6: Gray cells corresponds to the ideal correspondences, and bordered are the real ones.

A reasonable choice for the starting match is the pair of points with the lowest distance according to the distance matrix. Unfortunately, as said, this measure is ambiguous, so we might have many pairs with very low distances which do not represent good matches. Some additional features of the point can be measured in order to ensure a good initial match, such as the distance or orientation of the point with respect to the centroid of the contour. However, none of them are unambiguous, and pose extra assumptions on the shape of the hand.

For this reason a different approach was taken. Assuming temporal continuity between consecutive image frames, we can consider that the best match from the previous frame is a good starting pair for the current frame. In the first frame, the starting point is selected based on the distance matrix. Although the matching can be wrong in some points, there is a region between the alignment portion in the beginning and the end, see Figure 2.6, where the points are well matched. With this procedure, the selection of starting point is fast and accurate and the problems may occur only in the first frame.

2.3.3.3 Accumulated distance computation

Once the starting point pair has been selected, we can start the computation of the accumulated cost until each possible pair of points has been visited. The allowed transitions and their related costs J have to be defined. There are different possibilities related to the allowed transitions in a DTW system. We describe below what each of the possible transitions means:

- From pair (m, n) to pair $(m + 1, n)$: we advance one point in the first contour but stay in the original point on the second contour. We denote this an "alignment" jump.
- From pair (m, n) to pair $(m + 1, n + 1)$: we advance one point on both contours. We denote this a "matching" jump.
- From pair (m, n) to pair $(m+2, n)$: we advance two points in the first contour but stay in the original point on the second contour.
- From pair (m, n) to pair $(m + 2, n + 2)$: we advance two points on both contours.

In our system, we only allow the transitions from (m, n) to $(m, n + 1)$, $(m + 1, n)$ (alignment) and $(m + 1, n + 1)$ (matching). We have tested the performance of the system by allowing longer transitions and we concluded that the improvement was not significant enough to pay off the additional computational cost.

In the proposed approach, the cost associated to a transition serves as a multiplier of the distance associated with a point pair. It can be used to favor shorter transitions, or to favor "matching" transitions. For example, if there are no occlusions, we would want to favor matching transitions where we advance one point in both contours, and not transitions for alignment, where only one contour advances for one point. However, we experienced that the behavior with this approach is worse in cases of partial occlusions, where a lot of alignment transitions are required to match the contours properly. A good balance in our system was a factor of 1.5 for alignment, and a factor of 1 for matching, meaning that alignment has an additional cost. This improved considerably the matching process in the fingertips, where there are points with similar distances very close.

Given the transition costs J and the intrinsic costs (distance matrix) c, we can recursively compute the set of possible predecessors S_{pre}, the best predecessor (m, n) and the accumulated cost matrix C as follows:

$$
\begin{aligned}
S_{\text{pre}}(i,j) &= \{(i-1,j),(i,j-1),(i-1,j-1)\}\\
(m,n) &= \underset{(m,n)\in S_{\text{pre}}(i,j)}{\operatorname{argmin}} C(m,n)\\
C(i,j) &= C(m,n) + J(i-m,j-n)\times c(i,j) \qquad (2.8)\\
J(a,b) &= \begin{cases} 1 & \text{if } a=b=1,\\ 1.5 & \text{if } a+b=1,\\ \infty & \text{otherwise} \end{cases}
\end{aligned}
$$

An example with constant cost for the different transitions J ($J(1,0) = J(0,1) = J(1,1) = 1$) is shown in Figure 2.7. In order to compute the accumulated cost in the white bordered cell $(4,4)$, we add the intrinsic cost of the pair $c(4,4) = 1$ and the minimum accumulated cost for the possible predecessors, that in this case is an alignment transition from $(4,3)$. This accumulated cost is calculated for all the point pairs until the end point pair is reached.

1	5|26	7|22	12	1
4	13|21	9|15	**1|(1+3)**	17
3	8|8	5|6	2|3	15|18
2	**0|0**	1|1	7|8	11|19
	2	3	4	1

Figure 2.7: Distance matrix with accumulated costs. Each cell shows the intrinsic cost $c(i,j)$ and, if calculated, the accumulated cost $C(i,j)$

2.3.3.4 Backtracking

Once all the accumulated distances have been calculated, the backtracking process begins. The set of best correspondences is extracted by backtracking the "best predecessor" from the end point pair and repeating the process until the starting point pair is reached.

$$
(i_k, j_k) = \begin{cases} (K,K) & \text{if } k = K\\ \underset{(i,j)\in S_{\text{pre}}(i_{k+1},j_{k+1})}{\operatorname{argmin}} C(i,j) & \text{otherwise} \end{cases}
$$

2.3.4 Experimental evaluation

We compared the performance of the ICP algorithm developed in [23] and the DTW algorithm developed proposed here in a number of hand gesture and object manipulation sequences. For the ICP algorithm, we allow the maximum number of iterations to be

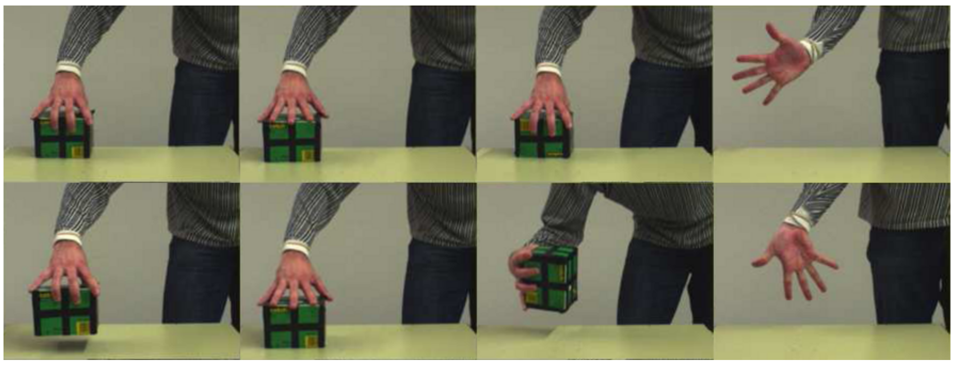


Figure 2.8: Two frames of each sequence used in evaluation: moving, pushing, rotating an object and simple hand waving.

50. The sequences on which the performance of the algorithms was evaluated included moving, pushing and rotating an object, and simple hand waving, see Figure 2.8. Ground truth was provided by manually marking the matched pairs in all frames. The results are represented by plotting the error probability density function.

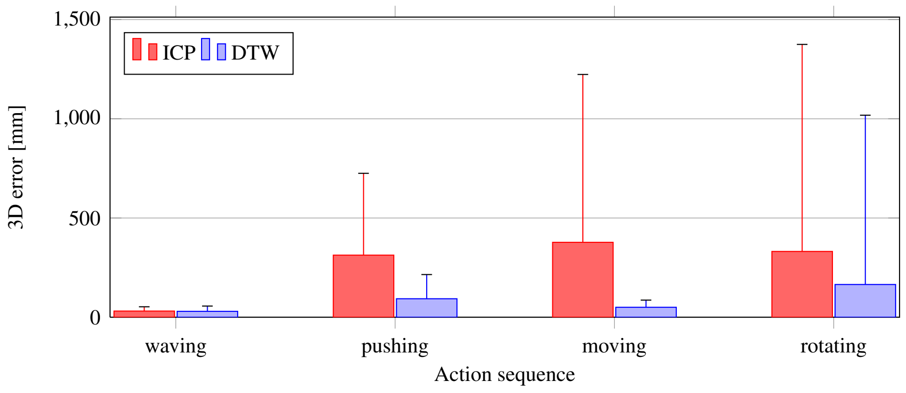


Figure 2.9: Error in fingertip location using DTW and ICP for four different sequences

The waving sequence is the simplest one. The hand contour conforms a planar surface parallel to the camera plane. These are the ideal conditions of the ICP algorithm, and the performance of this algorithm in this sequence is very good. The performance of DTW in this sequence is slightly worse, see Figure 2.9.

The rest of the sequences contain object manipulation actions, and the hand is partially occluded in some frames. ICP is less robust to occlusions than DTW, since usually the hand contour shape is considerably different in stereo images when there are occlusions present, see Figure 2.5. But the main advantage of DTW is that it shows a good performance for the case when the hand contour points do not lie on a plane, see Figure 2.9. This figure represents the mean error and standard deviation of the distance between the extracted fingertips and the ground truth.

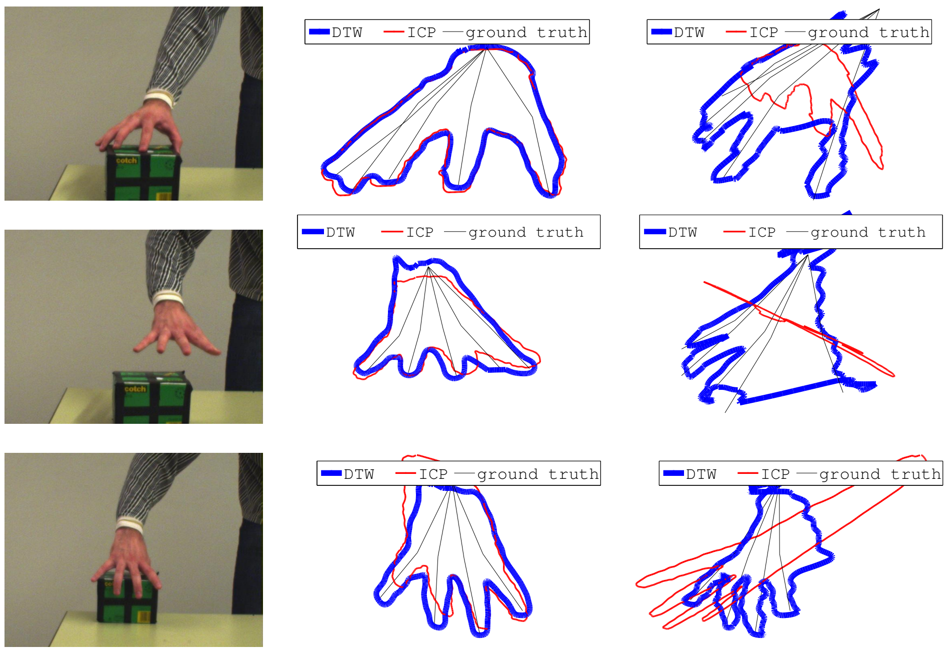


Figure 2.10: Pushing, moving and rotating sequences 3D border from the camera viewpoint and from a sideview

We have also performed a qualitative evaluation of the two methods. Some of the sample results are presented in Figures 2.10. Here, the reconstruction of hand contours performed with DTW and ICP is compared to the ground truth which is represented as a line skeleton from the wrist to the fingertips. In the first row of Figure 2.10, it is visible that the performance of DTW is better than ICP when the planar assumption is not satisfied anymore. Even if the extracted contours are almost the same from the camera viewpoint, the reconstructed 3D contour is clearly much better in the case of DTW approach. Second and third row of Figure 2.10 also show that the DTW approach not only outperforms ICP but also performs well in cases of occlusions. While the thumb is occluded by the rest of the fingers, it is partially visible in the 3D reconstruction since the right camera image (not present in the figure) had a better angle to visualize the thumb.

2.3.5 High Level Feature Extraction

The goal of the system presented in this section is to provide low level features (i.e. 3D silhouettes) to another module which computes higher level features (i.e. fingertip position) from them. Our initial idea was to compute the fingertips by detecting local maxima of the silhouette's curvature, similar to the method used in [24]. However, this method, which has been successfully used for controlling mouse gestures ([24]), fails when the hand movements observed are not so constrained. Based on [24], we computed the curvature of the hand silhouette as:

$$C(P) = \begin{cases} \frac{P_1P \cdot P_2P}{\|P_1P\|\|P_2P\|} & \text{if } \frac{\|P_cP_1\|+\|P_cP_2\|}{2} < \|P_cP\| \\ -\infty & \text{else} \end{cases} \tag{2.9}$$

where P_1 and P_2 are points from the contour separated by 30 points from P. The more parallel P_1 and P_2 are, the higher is the curvature and $C(P)$ is closer to one. When the contour is flat, P_1 and P_2 will be parallel but with opposite directions, so $C(P)$ is close to -1. The condition in Eq. 2.9 discriminates between convex and concave curvatures, giving an infinite value to concave parts of the silhouette like the finger bases. A point is considered a fingertip if its curvature is among the 5 lowest local minima.

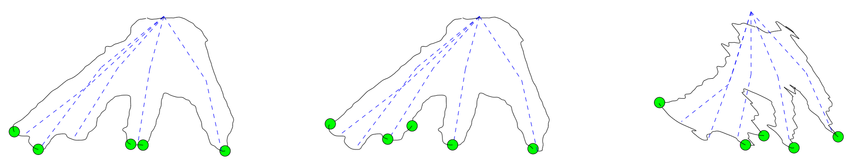

(a) 2D Fingertip extraction (b) 3D Fingertip extraction (c) 3D Fingertip extraction, side view

Figure 2.11: Fingertip detection based on silhouette curvature

In Figure 2.11 we can see that fingertip detection following this scheme is not robust either performing the curvature measurements in 2D (Figure 2.11a) or 3D (Figure 2.11b). In 2D the curvature can have multiple local minima within one finger which triggers multiple detection in a single fingertip. In 3D, the noise in the edge extraction results in spurious fingertips, even after smoothing the contour.

Even if we consider the fingertip extraction problem solved, extracting the joint angles from fingertip position is far from being trivial. In [151] this is achieved by using a neural network and fixing the position of the palm. This solution is however not easily extensible to a scenario in which the palm position is not fixed. If we consider a simplified model of the fingers which have 3 degrees of freedom (see Figure 2.12) there exists an closed form solution for the finger joint angles provided the 3D positions of fingertip and fingerbases [99]. The fingerbases could be estimated by either locating the hand palm plane or by curvature measurements, but these methods are subject to problems similar to the ones stated before. Furthermore, the closed form solution is not robust to noise, which is a potential problem given the quality of the reconstructed 3D contour.

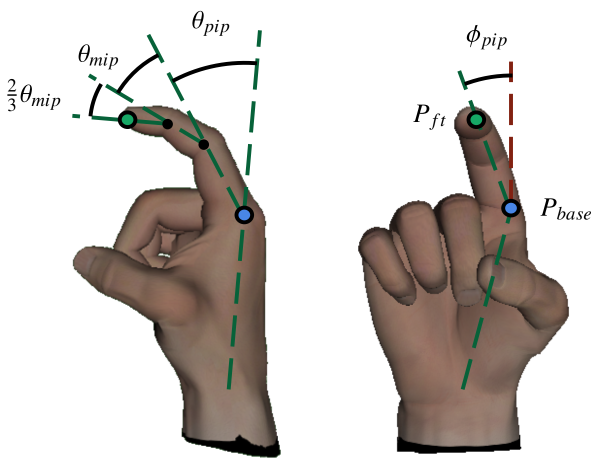


Figure 2.12: Simplified finger model with 3 degrees-of-freedom. Modeling fingers in this way makes possible a closed-form solution for the inverse kinematics of the hand.

All those reasons brought us to abandon this approach. Hand pose estimation based on low level features alleviate most of the disadvantages described above. The computational core is therefore moved from feature extraction to feature processing; however, the computational requires are moderate in discriminative approaches, as will be described in Section 2.4.

2.3.6 Summary

In this section we studied the problem of 3D hand contour extraction. We developed a method for matching 2D contours that improved state-of-the-art methods in the context of 3D hand contour matching. This was done to enable methods that might allow us to perform hand tracking based on these contours.

However, it is being shown that the robustness of fingertip extraction based on these 3D contours is low, and that the methods for further processing fingertip positions into full hand pose estimation rely on really accurate estimations that cannot be provided with the methods shown in this section.

Therefore we consider that hand pose estimation based on high level features such as fingertip positions is unreliable, and therefore the implementation shown here cannot be used as as the foundation of a hand pose estimation system. We will explore the direct usage of low level features in the next section.

2.4 Non-Parametric Hand Pose Estimation with Object Context

As it has been argued in Section 2.3, the extraction of high level features from hand images is sensitive to depth-extraction noise and, more importantly, occlusions. This lack of robustness encourages us to avoid extracting high level features for estimating the hand pose. Instead, we propose the use of low level features in a discriminative scheme that is real-time and robust to occlusions and segmentation errors. While the majority of methods in the literature assume a hand isolated from the surrounding environment, we argue that object occlusions can be beneficial for determining the hand pose in the context of object grasping. Our hand tracking method is non-parametric, performing a nearest neighbor search in a large database (100 000 entries) of hand poses with and without grasped objects. Temporal consistency in hand pose is taken into account, without explicitly tracking the hand in the high dimensional pose space. Experiments show the method to outperform other state of the art hand tracking methods.

2.4.1 Introduction

As already referred in Section 2.1, most of the hand pose systems consider a free hand isolated from the scene ([66, 200, 201, 214, 26, 190]). This assumption might be acceptable in scenarios such as sign language recognition, but not in an object grasping task. One of the few systems dealing with object occlusions is [100], who describes a generative model-based tracker that allows for objects in the hand. The large object occlusion makes reconstruction of a hand grasping an object in many ways a much more challenging task than reconstructing a free hand. In [100] the authors show that the hand pose can be reconstructed robustly *despite* the object occlusion, using a generative approach as illustrated in Fig. 2.13, top row.

In contrast to [100], we argue that knowledge about object shape gives important cues about the configuration of palm and fingers in contact with the object. Firstly, the object surface constrains the hand pose physically, assuming hand-object contact. Secondly, there is a functional correlation between object shapes and the manner in which they are grasped by a hand [11].

Neither the hand nor the object are explicitly reconstructed; the hand and the object are instead modeled together, encoding the correlations between hand pose and object shape in a non-parametric fashion. In spirit of recent methods for contextual recognition and estimation, e.g. [7], [220], the object occlusion thereby *helps* in the hand pose reconstruction.

Fig. 2.13, bottom row illustrates our approach. In our non-parametric method, pose estimation essentially corresponds to matching an observed hand to a very large database (100 000 entries) of hand views. Each instance in the database describes the articulation and the orientation of the hand. The configuration of a new (real) image can then be found using an approximate nearest neighbor approach, taking previous configurations into account.

Other hand pose estimation systems have used databases of hand views [214, 26, 190]. As explained before, the main difference between our system and the rest is the usage of contextual information in the form of hand occlusions to improve the hand pose estima-

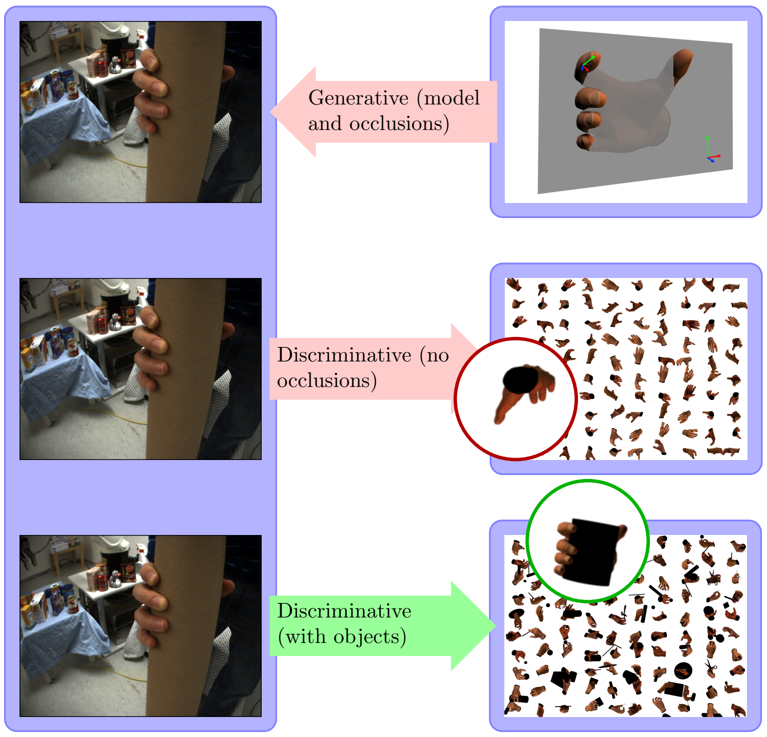

Generative (model and occlusions)
Discriminative (no occlusions)
Discriminative (with objects)

Figure 2.13: Hand pose estimation is traditionally approached in two different manners, either with a generative model (top) or using a discriminative approach (middle). A generative model tries to match a parametric model of a hand to the image evidence while the discriminative approach tries to model, parametrically or non-parametrically, a mapping from image to pose. Both approaches have significant problems estimating the hand pose in scenarios where large portions of the hand are occluded. This renders the image evidence too limited making both the generative and discriminative mapping ambiguous. Our method (bottom) takes a different approach and exploits contextual information in the scene such as object-hand interaction. Due to this we can reliably predict pose in scenarios with significant occlusion. We would like to point out that our model is not limited to scenarios where an object is being manipulated but equally valid to estimate a free hand.

tion. None of the three previously mentioned systems mentioned how to handle or take advantage from occlusions, and the experiments showed hands moving freely without any object occlusion. In [214], the application of a specially designed glove circumvents several problems associated with hand-pose estimation, making the problem as well as the approaches significantly different. The system described in [26] performs classification of human hand poses against a database of 26 basic shapes. They explicitly mentioned that their method "can only handle a few tens of hand shapes". In the other hand, our method aims to perform continuous hand pose estimation rather than isolated single-frame pose classification. The work from [190] can be regarded as the most similar to our work from the three methods described here. However, apart from the lack of contextual information in their model, they have to do simplifications in order to cope with the complexity of their approach, like treating independently hand rotation and hand articulation. Finally, both [190] (3 seconds per image) and [26] (15 seconds per image) run at a frame rate between one and two orders of magnitude lower than our algorithm.

In our system, the database contains hands both with and without grasped objects. The database depicts grasping hands including occlusion from objects with a shape *typical for this kind of grasp*; this encodes functional correlations between object shape and the articulation of the grasping hand. The occlusion shape is strongly correlated to grasping type which further has a strong dependency with the hand articulation. Since the underlying assumption is that appearance similarity can be related to similarity in hand pose the object shape contributes to the hand pose estimation.

In many scenarios it is hard to differentiate between the palmar and the dorsal ("backhand") sides of the hand. However, the object is much more likely to occlude the palm rather than the dorsal side. This is an example of how object knowledge can be exploited in order to resolve the ambiguities typically associated with hand pose estimation.

The probabilistic estimation framework is outlined in Sec. 2.4.2. The non-parametric hand model is described in Sec. 2.4.3, while Sec. 2.4.4 describes how inference is done over this model. Experiments in Sec. 2.4.5 show the present method to outperform other state of the art hand tracking methods. We also show qualitative reconstruction results for a number of synthetic and real test sequences.

2.4.2 Probabilistic Framework

We begin by explaining the notation used throughout the section. At a specific time instant t, let $\mathbf{x}_t$ be the articulated hand pose and $\mathbf{y}_t$ the corresponding image observation.

Given a specific image observation $\mathbf{y}_t$ we wish to recover the associated pose parameters $\mathbf{x}_t$ generating the visual evidence. Formally we will refer to the relationship between the pose and the image space as the generative mapping f,

$$\mathbf{y}_i = f(\mathbf{x}_i). \tag{2.10}$$

The naïve approach to infer the pose from an image would be to model the inverse of the generative mapping as a function using a regression model as in [16]. However, this is known to be a severely ill-conditioned problem as the image features are ambiguous implying the inverse mapping to be multimodal [3] as the generative mapping is not a bijection. In order to address this problem, several different approaches have been suggested: generative models [75, 100] which directly models f, approaches which rely on multiple views such as [66], or methods that exploit the temporal continuity in pose [75, 209].

In this section, our objective is to create a highly efficient method for situations where model-based generative approaches are inapplicable due to their computational complexity. Further, multiple views are not available in most applications, thus we take the latter approach and exploit temporal continuity to disambiguate the pose. We assume the pose space to be Markovian of order one, i.e that $\mathbf{x}_t$ depends only on the pose at the previous

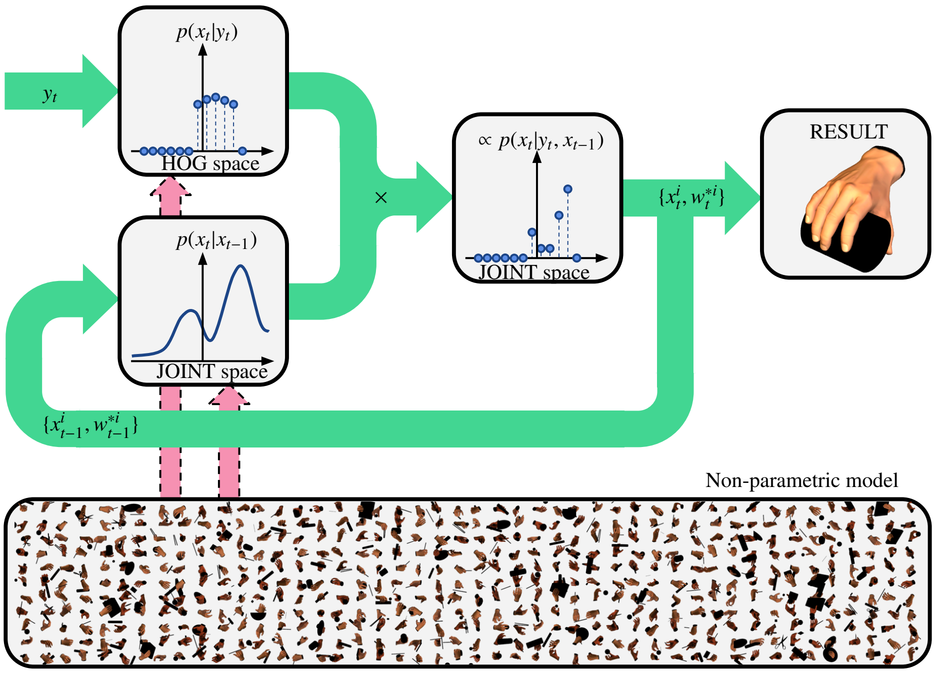


Figure 2.14: Schematic figure of the non-parametric temporal pose estimation framework. Given an image observation $\mathbf{y}_t$ a set of pose hypothesis $\mathbf{X}_t$ is drawn from the model and associated with a likelihood. Each of the hypothesis is given a temporal likelihood based on consistency with the hypothesis in the previous frame. The final estimate is the pose associated with the largest probability.

time step $\mathbf{x}_{t-1}$. The estimation task thus reduces to maximizing $p(\mathbf{x}_t|\mathbf{y}_t, \mathbf{x}_{t-1})$, which we decompose into an observation and a temporal model,

$$p(\mathbf{x}_t|\mathbf{y}_t, \mathbf{x}_{t-1}) \propto p(\mathbf{x}_t|\mathbf{y}_t)p(\mathbf{x}_t|\mathbf{x}_{t-1}). \tag{2.11}$$

As shown in Figure 2.14, this is solved with a non-parametric approach that models implicitly the appearance and temporal likelihood by means of a large database of images and their corresponding poses. To perform inference we use a truncated approach where we approximate the distributions in Eq. 2.11 using local models.

2.4.3 Non-parametric Representation

The first ingredient in our non-parametric hand pose estimation system is a training data set that can be assumed to "well" represent the problem. Generating such database poses a challenge as it needs to describe the variations in pose and image appearance at a sufficient resolution in order to make accurate pose estimation possible. However, with recent

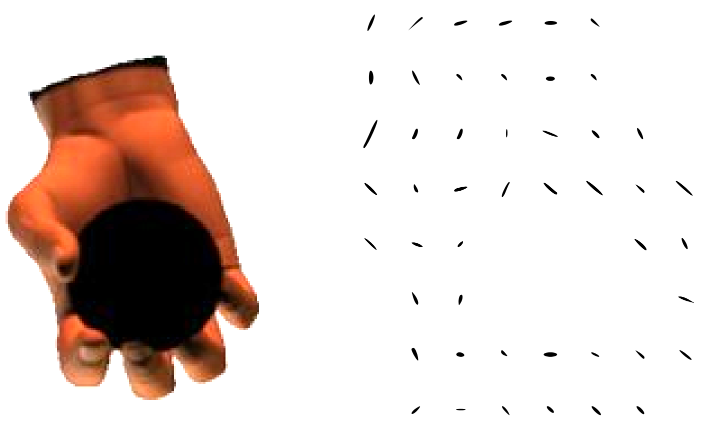


Figure 2.15: The left image shows an example from the database. The right image shows the associated image feature descriptor **y**. Prior to extracting the feature descriptor the object is segmented from the image, resulting in a "hole" at the corresponding position in the descriptor. This encodes the correlation between pose and object in a more robust manner compared to if the internal edges of the object would also contribute to the descriptor.

advances in Computer Graphics we can use the software package Poser which is capable of generating high-quality images of hands efficiently. Acquiring large sets of training data using this approach is not new and has proved to be very successful for pose estimation [16, 182].

The composition of the database is motivated by our research aim: understanding human interaction with objects, [76, 77, 7]. We select 33 different grasping actions according to the taxonomy presented in [11]. Further, each action is applied to a set of basic object shapes on which the grasp would naturally be applied. Each action is then discretized into 5 different time-steps. In order to make our approach view-independent we generate samples of each instance from 648 different view-points uniformly located on the view-sphere. This results in a database of over 100 000 instances (see, e.g., Fig. 2.15, left), which we assume samples the problem domain well.

2.4.3.1 Data Collection

The raw representation of an image (i.e. a mere collection of three-dimensional pixels) is very high dimensional, containing very different types of information: hue, brightness, shapes, etc. A large part of this information is irrelevant for our goal, which is estimating the hand pose. Moreover, the existence of such extra information makes both processing and storage of large amounts of images infeasible due to their large dimensionality. In this section we apply a two stage feature extraction approach with the aim of obtaining a compact representation of the image which keeps the hand pose information.

In the first stage the hand is segmented from the image. This task has been an active topic in computer vision research [65, 153, 168, 218, 224, 38, 176, 181, 187, 185]. Some systems use parametric models of the skin color [65, 153, 168, 218, 224] while other systems model the skin color non-parametrically with histograms [176, 181, 187, 185, 22]. Another choice to be considered for this systems is which color space should be used; the most popular spaces are normalized RGB [153, 218, 181, 187, 22] and HSV [168, 176, 224, 219]. Since our focus is not on skin-color segmentation, we used a very simple model, consisting in a range of colors in HSV space. When a pixel falls inside such range of colors, is considered skin. HSV color space was chosen since it has been reported to outperform other spaces [199]. This process also removes the object being grasped, since we are not handling skin-colored objects.

After the image pixels are classified as skin or non-skin, they are grouped into connected components. We find all the blobs with a pixel area over a threshold and pick the one closest to the bottom-left corner, in order to reject head and left hand.

Once the hand is extracted from the image, we compute features from it in order to reduce its dimensionality. A large amount of work within Computer Vision has been focused on developing different image features [117, 133, 143]. Our ideal image feature should be robust to segmentation errors, sensitive to non-textured regions and fast to compute. We considered two widely used features for our problem: Histogram of Oriented Gradients (HOG [64]) and features based on edges ([41]).

Histogram of Oriented Gradients (HOG)

Histogram of Oriented Gradients computes the gradient orientation of a single channel image (in this case our original image converted to grayscale) in each pixel, and groups these orientations into histograms for groups of neighboring pixels. Gradient orientation $\Phi \in [0, \pi)$ is computed from the segmented hand image H as

$$\Phi = \arctan(\frac{\partial H}{\partial y} / \frac{\partial H}{\partial x}) \tag{2.12}$$

where x denotes downward (vertical) direction and y rightward (horizontal) direction in the image.

From Φ, a pyramid with L levels of histograms with different spatial resolutions are created; on each level l, the gradient orientation image is divided into $2^{L-l} \times 2^{L-l}$ equal partitions. A histogram with B bins are computed from each partition. Sometimes, like in our case, the multiple scales used in the HOG do not improve the performance of the feature, and only the last level L is used. An example of histograms used in our system can be seen in Figure 2.15.

The hand view is represented by y_t which is the concatenation of the histograms at all levels in the pyramid. The length of y_t is thus $B \sum_{l=1}^{L} 2^{2(L-l)}$. For the particular case in which only the lowest level of the pyramid is used, the length is $2^{2(L-1)}B$.

In Figure 2.16 we can see a comparison of the performance of HOG with different parameters. Only the entry hog8x8x8pyr uses the full pyramid; the rest only use the last level. The name follows the notation hog$2^{2(L-1)} \times 2^{2(L-1)} \times B$. We can observe that hog8x8x8 gives the best performance with smaller size than hog8x8x8pyr and hog16x16x8.

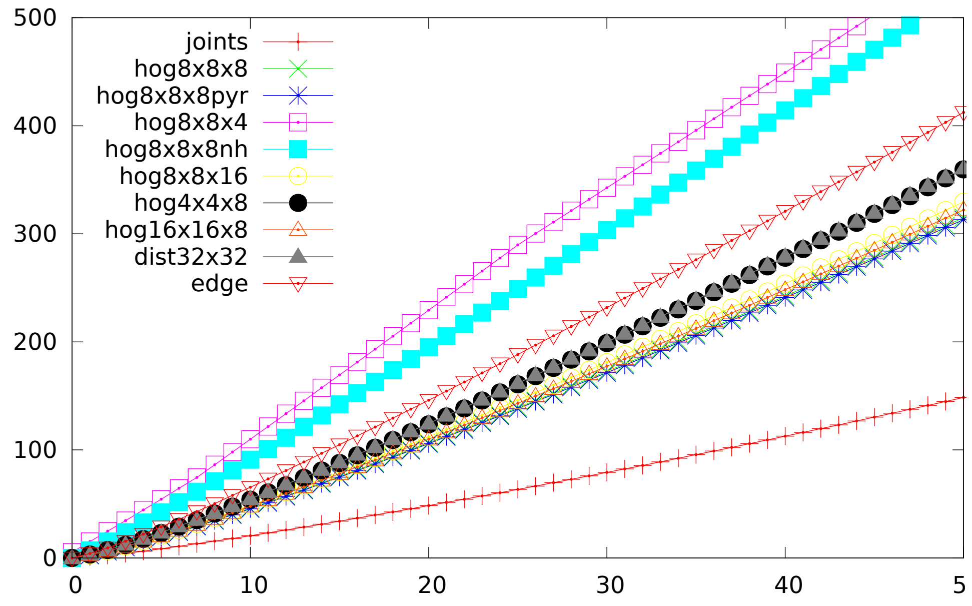


Figure 2.16: Cumulative mean pose error of exact nearest neighbor (0 to 50) for a set of different image features. The result is computed by averaging the error of querying each feature in the database. Joints is the error in pose space being a lower-bound on the error and shows the precision in our database. hogA×A×B symbolises a hog of an image divided into A×A non overlapping cells, where the histogram was performed with B bins (see Fig 2.15 for an example of a $8 \times 8 \times 8$). pyr means the HOG feature included lower resolution cells (1×1,2×2,...$A \times A$). nh means normalized holes: the histogram is normalized to sum one (this creates problems in "holes", i.e. when the image presents small amount of gradients in a cell). In dist32×32 each image is represented by their distance transform subsampled to 32×32 pixels. edge is not based on the mean square error between features, but the chamfer distance between images. The trend of the curves in the figure is an indication of the smoothness that can be associated with each feature representation.

Edge-based features

Edge extraction is another popular feature widely used in computer vision [53]. Edges are related to gradients since its computation is usually based on finding local maxima in the gradient image Φ, followed optionally by some filtering processes such as hysteresis-thresholding ([53]).

In our system we are interested in features which can be used for comparing different images. This represents an instance of the shape matching problem. There are two approaches towards shape matching with edges: the algorithms which require correspondences between edges (such as Iterative Closest Point [35]) and the methods which do not

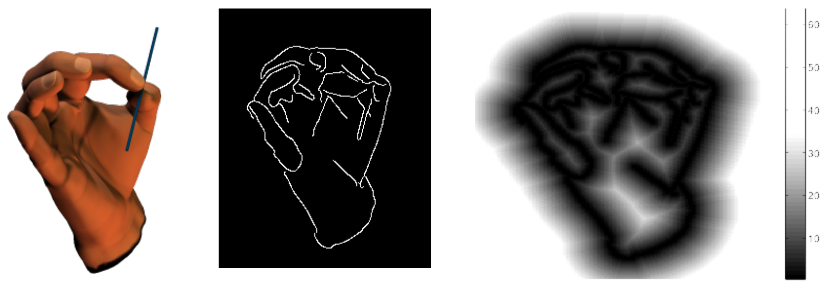


Figure 2.17: Hand image, edges and distance transform

require correspondences (such as Chamfer matching [46]). We have already described that the matching problem for human hand data is not trivial in Section 2.3, and therefore we will use methods which do not require initial correspondences.

Chamfer distance measures the distance between edge points in one image and the closest edge point from the other image.

The sets of points $\mathcal{H}, \hat{\mathcal{H}}$ considered are usually edges extracted with a Sobel or Canny operator from images $H, \hat{H}$.

$$d_{cham}(\hat{H}) = \sum_{\hat{h} \in \hat{\mathcal{H}}} \min_{h \in \mathcal{H}} ||h - \hat{h}|| \tag{2.13}$$

In our system, one real image H will be compared with several images from the database $\hat{H}$. Therefore it makes sense to precompute the distance from each pixel to the closest edge for the real image.That is the so called distance transform (Figure 2.17) [41]:

$$d_{transf}(p) = \min_{h \in \mathcal{H}} ||p - h||, p \in H \tag{2.14}$$

$$d_{cham}(\hat{H}) = \sum_{\hat{h} \in \hat{\mathcal{H}}} d_{transf}(\hat{h}) \tag{2.15}$$

Once we compute d_{transf}, we can compute $d_{cham}(\hat{h})$ multiple times for different images rather fast ($O(n)$ with the number of edge points). Therefore, we can compute the distance transform d_{transf} of the real image and then check how similar are the images from the database in linear time with the number of edge points of each virtual image.

The Chamfer distance between images as described in Equation 2.15 is relatively slow compared to features which can be compared in an Euclidean fashion, i.e. considering the feature a high dimensional vector and measuring distance between features as the Euclidean norm of the vector's difference. Therefore, we also tried comparing the distance transform of images in an Euclidean fashion.

Figure 2.16 exposes the performance of the described features compared to the HOG features. The entry edge represents the chamfer distance entry, while dist32 × 32 corresponds to an Euclidean comparison between the distance transform of the image, subsampled to 32×32 pixels to make the computation time comparable to the one of HOGs. The graph shows that both features are worse than the best HOG (hog$8x8x8$), and chamfer distance performing worse than the distance transform Euclidean distance. For this reason the feature used in the remaining of this section will be hog$8x8x8$, which has a dimensionality of 512. The appearance of this feature can be observed in Figure 2.15, right.

Object processing

Our approach also exploits contextual information of the grasped object when estimating the hand pose. The object contains a significant amount of information about the pose (and vice versa). In a learning based framework, which assumes having a training data set which describes the problem domain well, the natural inclination is that the model would be limited to handle objects which are included in the database. Such a model would have to be of a size that would render it infeasible to use. However, in our model the object is removed. This means the occluding shape of the object effects the representation while the internal edges of the object do not, see Fig. 2.15. This representation can robustly be extracted from the image and is capable of generalizing over different objects. As we will show in the experimental section this sufficiently models the correlation between hand and object allowing estimation in scenarios of severe occlusion.

Having aquired a low-dimensional efficient representation of the image, as described above, the database is completed by associating each image with its corresponding pose parameters. The pose vector $\mathbf{x}$ is composed of the rotation matrix of the wrist w.r.t. the camera and the sines of the joint angles of the hand.

2.4.4 Inference

The conditional pdf over hand pose $\mathbf{x}_t$ is factorized into two different terms, an observation likelihood $p(\mathbf{x}_t|\mathbf{y}_t)$ and a temporal consistency model $p(\mathbf{x}_t|\mathbf{x}_{t-1})$, see Eq. 2.11. Below we detail the observation model , followed by the temporal model, and finally show how $\mathbf{x}_t$ is estimated.

2.4.4.1 Observation

The pdf $p(\mathbf{x}_t|\mathbf{y}_t)$ is approximated by indexing into the database of hand poses using the image representation $\mathbf{y}_t$, and retrieving the nearest neighbors in the space spanned by $\mathbf{Y}$. Each image feature $\mathbf{y}^j$ is associated with a pose $\mathbf{x}^j$. The poses corresponding to the kNN $\{\mathbf{y}_t^i\}$ can thus be retrieved. Together with the weights, they form the set $\{(\mathbf{x}_t^i, w_t^i)\}$ which is a sampled non-parametric approximation of $p(\mathbf{x}_t|\mathbf{y}_t)$.

An exact NN approach would be too computationally intensive therefore we consider approximative methods.

This problem was studied by Stenger et at. [190] through the construction of a hierarchical bayesian filter, which refines the likelihood of the observations given the possible states only on the state space hypervolumes which present a significant probability mass.

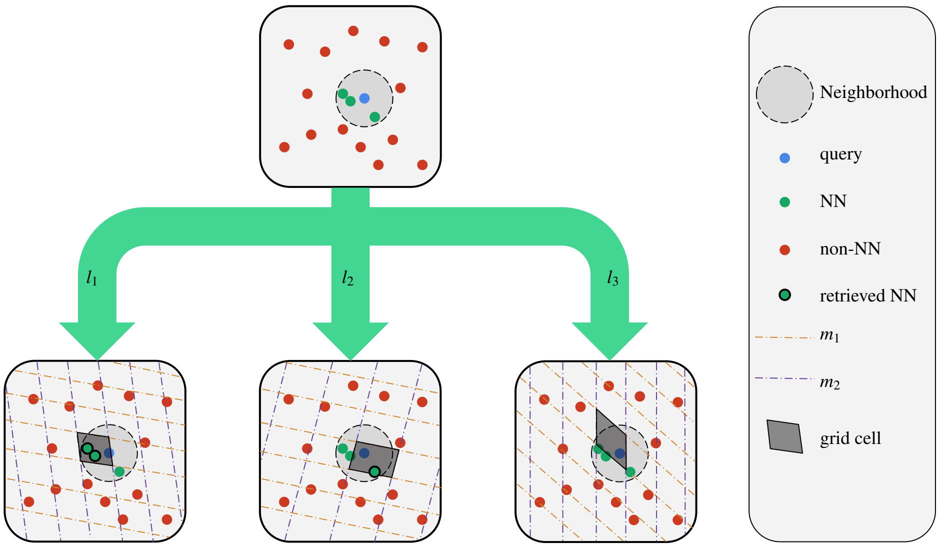


Figure 2.18: *Locality Sensitive Hashing methodology*

In a similar fashion, we pre-process our feature space in order to allow fast retrieval of nearest neighbors. Since we compare our features in an Euclidean fashion, several existing methods are already available to speed-up our approximate nearest neighbor retrieval. We compared Multi-Probe Locality Sensitive Hashing (Multi-Probe LSH) [71] and Fast Library for Approximate Nearest Neighbors (FLANN) [144], see Fig 2.19.

Multi-Probe Locally Sensitive Hashing (LSH)
A Locality Sensitive Hashing is a random mapping defined on a set of objects, such that the probability of two objects being mapped to the same value reflects the similarity between these objects [54]. Those values to which the objects map are lower dimensional than the objects themselves, and the evaluation of the mappings is fast. Therefore the comparison of the objects in the mapped space is much faster than in original space.

LSH creates a number L of hash tables (l_1,l_2 and l_3 in Figure 2.18) composed by M hash functions each (m_1 and m_2 in Figure 2.18, different for each hash table). Each hash table can be thought as a grid in the high dimensional space which defines locality in a different way. The grid is created by a set of M lines (the hash functions) which define the unit vector of each of the dimensions of the grid. The thickness of the grid is controlled by a parameter W.

$$l_i(v) = < m_1(v), m_2(v), \dots m_m(v) >, i = 1, 2 \dots L \tag{2.16}$$

$$m_j(v) = \lfloor \frac{a_j \cdot v + b_j}{W} \rfloor, j = 1, 2 \dots M \tag{2.17}$$

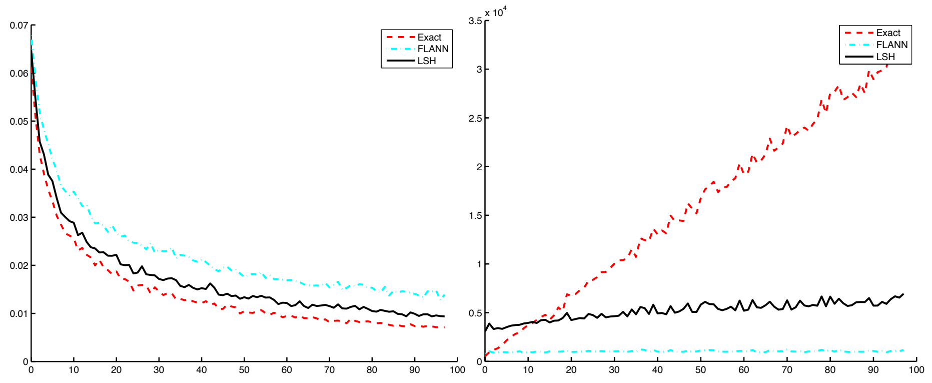


Figure 2.19: The plot shows the prediction error (left) and average query time (right) as a function of database size (as percentage of the training set) for finding the nearest neighbor in the database. Using 10% of the database for testing and the reminder for training. Two approximative methods, LSH and FLANN, are compared with an exhaustive search. LSH performs slightly better compared to FLANN but both approximates the exhaustive search well. The right plot shows the query time increasing linearly for the exhaustive search while the approximative methods being sublinear.

The parameters M and W control the locality sensitivity of the hash function. In order to obtain the nearest neighbors for a given query, the query is projected into each of the L grids (hash tables) and every training data which is projected into the same grid cell for any hash table is considered a nearest neighbor. See Figure 2.18 for a graphical representation of this.

In order to achieve good selectivity, the number of hash functions L should be large. In order to decrease such number, Multi-Probe LSH does not only test the grid cell where the query was mapped to, but also a number T of adjacent cells. Large values of T allow to achieve similar results with less hash tables, but more spurious results appear when T becomes too big.

We used the library LSHKIT for multi-probe LSH. This package optimizes the values of parameters L, M, W, T for a given dataset and required recall.

Hierarchical K-means Tree

Another popular approximate nearest neighbor approach is to create a hierarchical tree based on the training data and explore only the promising branches. The Fast Library for Approximate Nearest Neighbors (FLANN) library [144] implements a hierarchical k-means tree algorithm (apart from other algorithms) which has been shown to have a good accuracy with very fast query time. The tree is constructed by recursively splitting the points under each branch into K clusters using k-means clustering. Each of the clusters will conform a new branch which should be repartitioned recursively. In FLANN the tree is explored in a best-first way instead of depth-first.

Nearest Neighbors Methods Comparison
In Figure 2.19 we can see a comparison of the prediction error and query time using Multi-Probe LSH, FLANN and a naïve exact nearest neighbor approach. Based on the results, we decided to use LSH in our experiments as it showed an attractive trade-off between computational complexity and prediction accuracy.

2.4.4.2 Temporal Consistency

As described in Sec. 2.4.2, the temporal consistency constraint $p(\mathbf{x}_t|\mathbf{x}_{t-1})$ is modeled as a parametric function. It is used as a conditional prior to re-weight the sampled distribution $\{(\mathbf{x}_t^i, w_t^i)\}$, approximating $p(\mathbf{x}_t|\mathbf{y}_t)$. We assume that the global orientation (**view**) and the joint configuration (**joint**) of the hand to be independent. This leads to the following factorization of the temporal model,

$$p(\mathbf{x}_t|\mathbf{X}_{t-1}) = p(\mathbf{x}_t^{\text{view}}|\mathbf{X}_{t-1}^{\text{view}})p(\mathbf{x}_t^{\text{joint}}|\mathbf{X}_{t-1}^{\text{joint}}). \tag{2.18}$$

We assume that our model is getting observations densely enough in time such that the assumption of smooth variation in pose and view holds. The naïve modeling approach would thus be to penalise estimates by their deviation in pose space to the current time estimate. However, by only using the best estimate we are making the model unnecessarily sensitive to noise, which might result in considerable drift in the prediction. A more sensible approach would be to make use of all the hypotheses $\mathbf{X}_{t-1} = \{\mathbf{x}_{t-1}^1, \dots, \mathbf{x}_{t-1}^n\}$ in the previous time instance and propagate them through time. We can do so by modeling the conditional distribution $p(\mathbf{x}_t^{\text{view,joint}}|\mathbf{X}_{t-1}^{\text{view,joint}})$ using a kernel density estimation (KDE) approach [142] as a weighted combination of the hypotheses from the previous frame. This implies propagating a potentially multi-modal distribution in time, making the temporal model significantly more flexible and expressive, allowing us to resolve ambiguities. In specific we use a spherical Gaussian kernel placed on each of the 20 best poses from last frame, weighted according to the probability $p(x_t|x_{t-1}, y_t)$ for each of those poses. The covariance of the spherical Gaussian is determined empirically. The improvements of such a method compared to a single hypothesis model can be seen in [4].

The performance of this multi-modal modelling of the temporal distribution was compared against a simpler model in which

As we will show in Sec. 2.4.5, having a strong temporal model allows us to perform prediction in noisy scenarios where the image observations are uncertain.

2.4.4.3 Estimation

In order to get a pose estimate at a specific time instance t we wish to find the pose $\hat{\mathbf{x}}_t$ associated with the highest probability in the data-base,

$$\hat{\mathbf{x}}_t = \text{argmax}_{\mathbf{x}_t}\ p(\mathbf{x}_t|\mathbf{y}_t, \mathbf{X}_{t-1}) = p(\mathbf{x}_t|\mathbf{y}_t)p(\mathbf{x}_t|\mathbf{X}_{t-1}). \tag{2.19}$$

Since both observation and the temporal probability are expressed using a truncated local model the above maximization can be done exact by exhaustive evaluation over the full set of hypothesis.

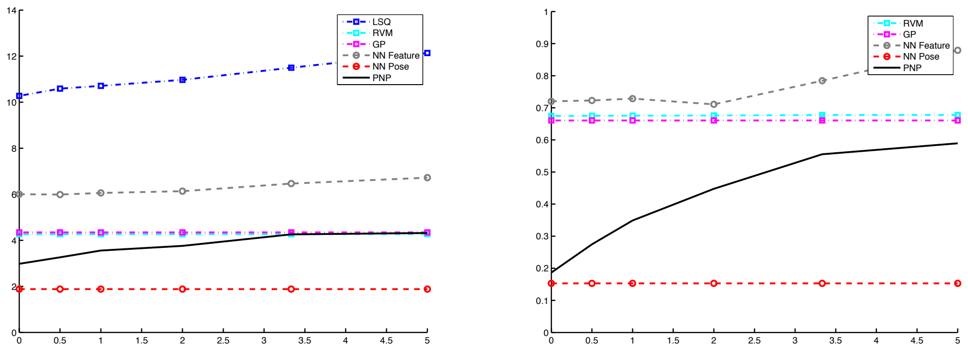


Figure 2.20: The left plot shows the average joint prediction error with increasing segmentation noise. As can be seen, the linear least square regression results in a very large error indicating that the relationship between feature and pose is strictly non-linear. The RVM and the GP are unable to model the mapping and do in fact always predict the same pose: the mean pose in the training data, irrespectable of image observation. In other words, this means that the appearance-to-pose mapping is underconstrained and does not take functional form. The three non-parametric approaches (NN Feature, NN Pose and PNP (our proposed model)) are capable to model in such scenarios. From the results we can see that an exact nearest neighbor in the feature space results in a worse result compared to the mean of the data while our approach performs significantly better – also indicating that the mapping is non-unique. The dashed red line shows the results of an exact nearest neighbor in the pose space and is therefore a lower-bound on the error of our method as it shows the resolution of the data-base. The norm in joint-space is not easily interpretable in terms of quality of the prediction as it does not respect the hierarchical structure of the hand, see Fig. 2.22. Therefore, the right plot shows the same mapping results, but with an error norm in terms of finger joint 3D positions. This shows even clearer how well our suggested method performs. With very little noise we are close to the exact NN lower bound, with increasing segmentation error asymptotically moving towards the mean. Note that 5% error corresponds to a very weak segmentation, see Fig. 2.21. Further, our approach significantly outperforms the exact nearest neighbor in feature space. This clearly indicates how important temporal information is in order to disambiguate the pose.

2.4.5 Experiments

We perform three sets of experiments using the proposed method. First we compare our approach to a baseline of other state of the art algorithms. In order to make an evaluation in terms of a quantitative error this experiment is performed using synthetic data where the joint configuration is known. Synthetic data also allows us to control the amount of noise in the images. Both our method and the baseline methods are evaluated in left robustness towards noise in the image observations. In the second set of experiments we evaluate our method in a qualitative manner on synthetic sequences with added image noise. The third set of experiments is performed on challenging real-world sequences.

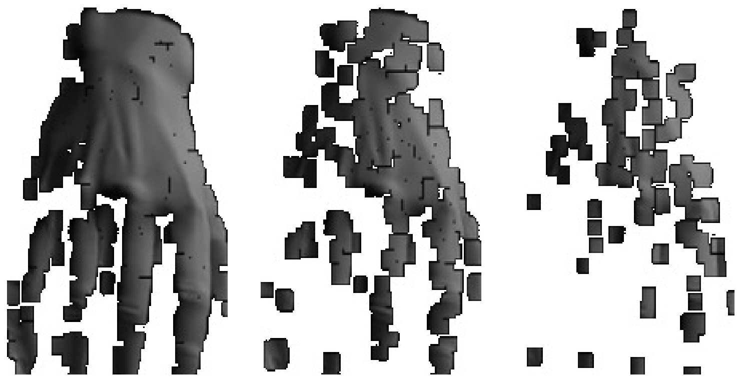

(a) α = 0.5% (b) α = 3.3% (c) α = 5%

Figure 2.21: Artificial corruption of the segmentation of the synthetic test data. The corruption is performed as follows: by first randomly removing α percentage of the pixels from the segmentation, a partial segmentation is created. This partial segmentation is then applied with the morphological operators of erosion and dilation in order to propagate the noise over the image. From left to right, examples of increasing segmentation noise are shown.

2.4.5.1 Baseline

We compare our method to a set of regression models. In specific, we use Least Square Linear Regression (LSQ), the Relevance Vector Machine (RVM) [204] and Gaussian Process regression (GP) [170] to model the mapping from input features to pose. Each of these models have previously, with significant success, been applied to pose estimation [16, 66, 221] for both hands and full body pose.

All above models are based on a fundamental assumption that the relationship between image and pose taking functional form; LSQ assumes linear form, while RVM and GP can model more flexible mappings. We compared these three methods to the suggested approach on 4 different synthetic sequences with varying degrees of added image noise, see Fig. 2.21, in the image observations, see Fig. 2.20. Neither the poses nor the objects in the test sequences were present in the database.

The results clearly show that the mapping from image features to pose is both highly non-linear and non-unique (multi-modal). This implies that it cannot be modeled using a functional approach. With respect to the results one could argue that we for completeness should compare with generative approaches which by design are capable of modelling in a multi-modal mapping scenario. However, in general, the computational complexity of applicable generative models are significantly higher.

Further, one could argue that a model based generative approach which minimizes a matching cost between the model and the image could be applicable. However, our motivation in this paper is to exploit hand and object information jointly. Building such a parametric model is associated with significant challenges: the only way to proceed with

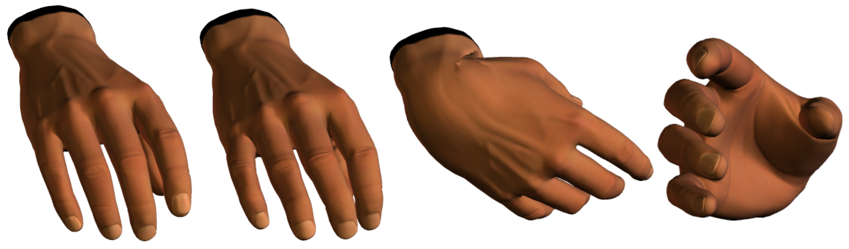


Figure 2.22: Four different hand-poses are shown. The left-most image corresponds to the ground truth pose and the remaining images are estimates of the ground-truth. The estimates are ordered according to decreasing joint angle error. This clearly exemplifies how badly joint angle error corresponds to the quality of the estimate. This is because the norm in joint space assumes each dimension to contribute equally to the quality of the prediction. Therefore it does not reflect the hierarchical structure of the hand where error higher up in the chain (such as in the last two examples) effects the position of every joint further down the chain compared to the first prediction where the errors are concentrated closer to the finger tips.

such a model is to independently model hand and object, which adds further computational complexity.

One possibility to proceed is then to reduce the data-base and use a much smaller set of training data. However, we argue that the problem is of such complexity that in order for it to be well represented it requires a data set of size which will not be computationally feasible for generative model to handle. Fig. 2.19 shows the error of NN regression as a function of increasing database size.

2.4.5.2 Synthetic Sequences

In order to evaluate the qualitative performance of our method in a controlled scenario, we applied the model to image sequences with a controlled noise level. The results are visualised in Fig. 2.23.

The estimated pose over the two sequences is accurate while the associated object varies. This implies that our assumption that objects generalize over pose and provide important contextual information is correct.

2.4.5.3 Real Sequences

In order to show the performance of our method in a real world manipulation scenario, we let three different subjects, two men and one woman, manipulate three different objects. The objects are not contained within the model. The results are shown in Fig. 2.24.

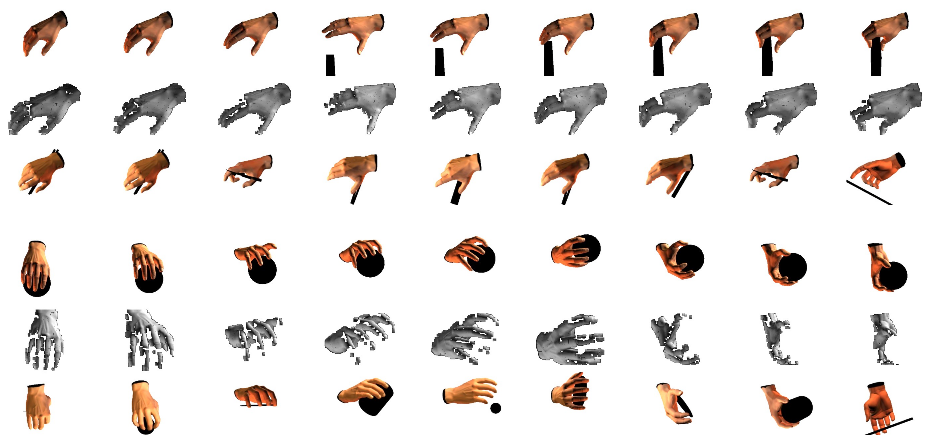


Figure 2.23: Qualitative results of our approach applied to synthetic data. The top and the forth row show the ground truth pose, the second and the fifth row show the segmentation from which the image features are computed. The segmentation has been corrupted by artificial noise with $\alpha = 0.5\%$ as explained in Fig. 2.21. The third and last row show the corresponding predictions from our system. The two grasping sequences are applied to two different objects, in the first sequence a book and in the second a ball. We show the predicted hand-pose but also the object that is associated with the specific pose in the database.

As can be seen from the results, our model is capable of accurately predicting the pose of the hand. In each of the sequences the test hand shape and appearance is different from the database hand model, while there is no observable degradation in performance, showing that our model is robust to different hands. Further, as neither of the manipulated objects are represented in the model this further supports the notion that grasp generalises over objects and that the objects' influence on the grasp provide important cues. This clearly shows that our system is capable of exploiting such information.

Many popular dynamical models that have been proposed to the problem of pose estimation are based on auto-regressive models [213], which assumes that the trajectory in time takes functional form. Even though our dynamical model is parametric, it is based on hypotheses from the non-parametric kNN model. This means that it is considerably more flexible and can recover from bad estimates in situations where an auto-regressive model will fail. To highlight this strength we tested our model to a set of highly challenging sequences with fast non-linear motion and significant occlusion. This results in significant errors in the visual features. In Fig. 2.25 the results clearly show the strength of our approach, as it is able to track in such scenarios.

Further, we would like highlight the efficiency of our algorithm. It runs in real-time which makes it applicable in many different scenarios where pose estimation is an important source of information.

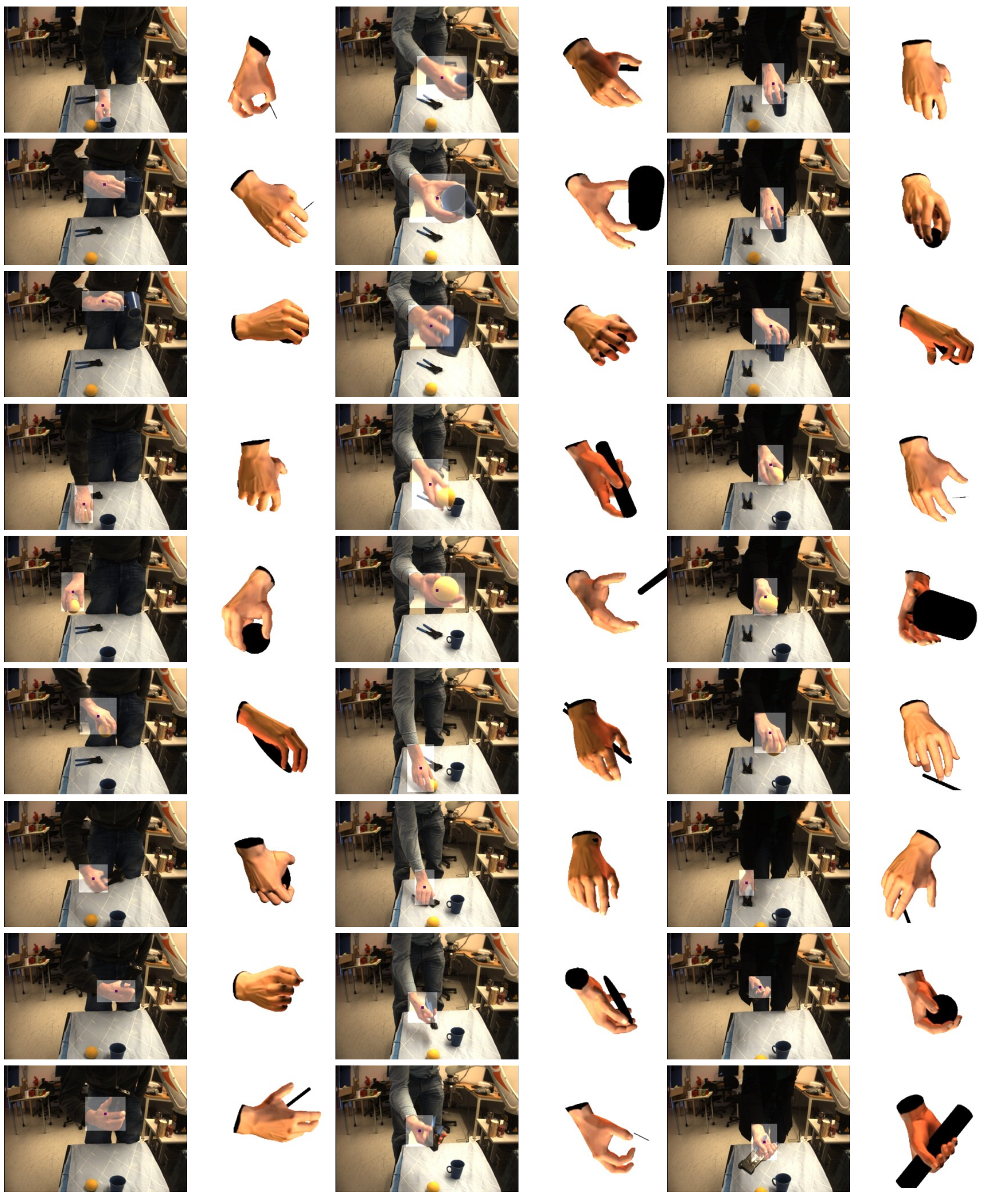


Figure 2.24: Estimations of real world sequences. The first column represents the input to the system for subject 1, the second column represents the most likely pose estimated for such input, and so on (up to three subjects). The rows represent three different frames of the manipulation of three different objects which do not exist in the database. As can be seen, the model correctly predicts the hand pose in each of the three different sequences.

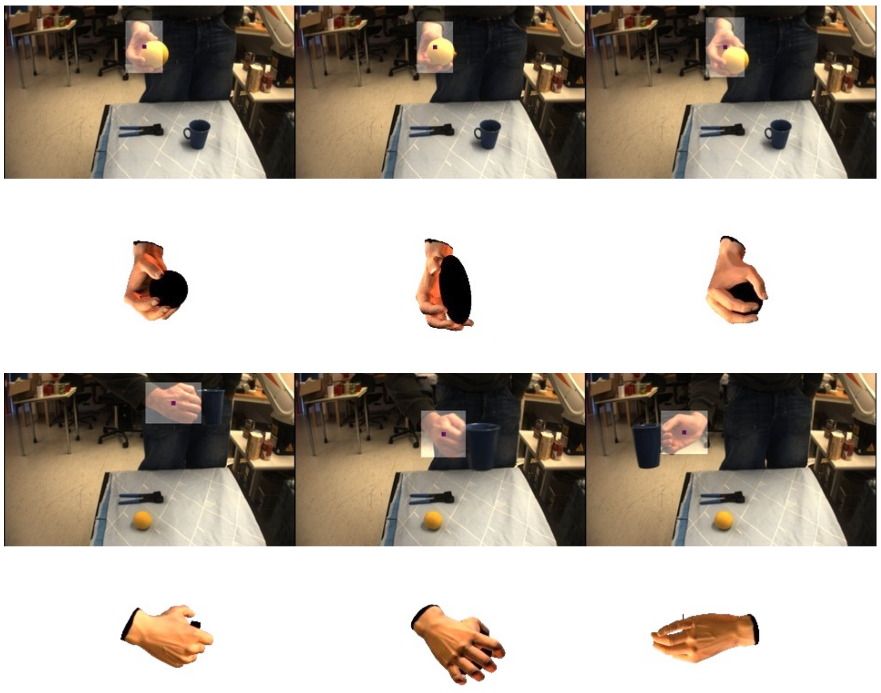


Figure 2.25: The above sequences shows two challenging examples. In the top sequence a significant portion of the hand is occluded by the object. However, our proposed method still manages to correctly estimate the pose of the hand. This clearly shows the strength of jointly estimating the object and the pose rather than seeing them as independent. The bottom sequence is an example where the subject manipulates the objects in a rapid fashion in a highly non-linear manner. In such scenarios most dynamical models commonly applied in pose estimation will over smooth the solution or be unable to predict at all due to being fundamentally auto-regressive approaches. Our model correctly predicts the pose in the two first frame while the last estimate is erroneous. This error is an implication of the Markov one assumption in our temporal model which thereby is not capable of modelling inertia and therefor is unable to resolve the ambiguity in the image sequence.

2.4.6 Summary

We presented a efficient non-parametric framework for full 3D hand pose estimation. We showed through extensive experimentation that the proposed model is capable of predicting the pose in highly challenging scenarios corrupted by significant noise or with rapid motions. Further, our model is efficient and runs in real-time on standard hardware.

The fundamental contribution is a system capable of exploiting contextual information in the scene from the interaction between the hand and a potential object. We show how this information can be exploited in a robust manner, making our system capable of generalizing the pose over different objects. This enables employing a fast discriminative method to scenarios where only expensive generative methods previously would have been applicable. We employ a multi-modal temporal model, allowing us to resolve ambiguities through temporal consistency. Our model could easily be extended to simultaneously estimate both the hand pose and the object shape by appending the inference scheme with a smoothness term with respect to object.

2.5 Discussion and future work

This Chapter has described two approaches towards estimating the human hand pose in the particular context of grasping actions. Section 2.3 we described the our work on 3D extraction of the hand 3D edges, and how such work might be extended towards a hand pose estimation system. However, as described in Section 2.3.5, such an approach has several robustness problems. In Section 2.4 we proposed an alternative based on a discriminative system. The system works in real time and it is robust to occlusions and segmentation errors. In the remaining of this Chapter we will present our future directions towards improving the performance of this system.

2.5.1 Learning a representation

In this section we have assumed that the Euclidean distance is a sensible similarity measure for the image features and hand poses. The nearest neighbor lookup for the appearance likelihood relies on the assumption that if two features are close according to the $\mathcal{L}_2$, they are "similar". Similarly, in the computation of the temporal likelihood there is an implicit assumption that if $||x_t^i - x_{t-1}||_{\mathcal{L}_2}$ is low, x_t^i is similar to the previous estimation and thus more likely. However, the $\mathcal{L}_2$ might not correspond to our idea of similarity for any of the previous cases.

As can be seen in Figure 2.22, proximity in $\mathcal{L}_2$ norms can be deceiving for joint angle spaces. This can be alleviated by using other spaces, like $3D$ positions of hand parts, by weighting different joints according to their level in the kinematic chain, or by whitening the variables according to their variance in training sets. However, better representations can be achieved provided training data composed by typical sequences that can appear in testing scenarios. Such a representation would have two requirements: (1) a small difference in joint configuration should corresponds to a similar pose, (2) poses that are likely to appear in sequence should be in close proximity. Learning a transformation of the data guided by a specific desired proximity relationship is known as metric learning [126]. We envision the optimization of a metric which have a trade-off between the original $\mathcal{L}_2$-norm proximity and the hints provided (as rankings in our case: this pose x_i should be closer to x_j than x_k) through the training data.

Given that our ultimate goal is to estimate the pose that can generate an image similar to the real one, it is sensible that the visual feature space mimics the hand pose space. Our goal for the image features representation should be to shape the space in such a way that

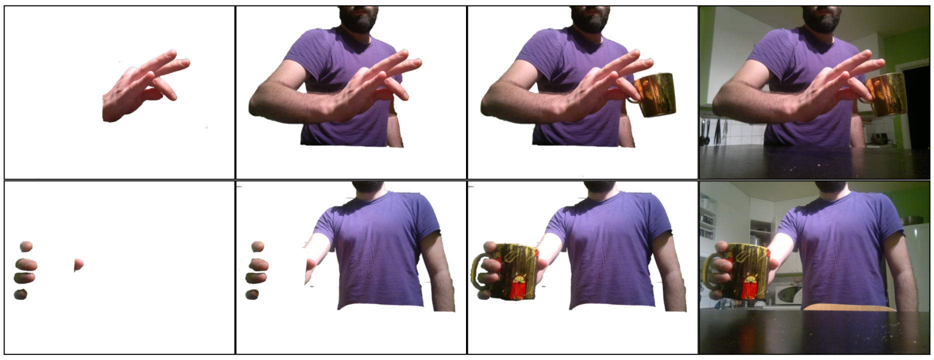


Figure 2.26: The perception of hand poses is largely influenced by its surrounding elements or context. The left-most column shows the information our system right now: only the segmented hand. However, the hand pose becomes much easier to identify when we see the rest of the body, mainly the arm. The existence of an object provide more details about the hand, e.g. the distance between fingers in the upper row, or the pose of the palm in the lower row. Finally the whole scene give us higher level knowledge, such us possible actions to be performed with the object.

proximity in image space resembles proximity in hand pose space. This can be represented as another instance of metric learning.

2.5.2 Hybrid Discriminative-Generative Hand Pose Estimation

Discriminative methods have as an advantage their speed and their possibility of exploring globally the sampled pose space. However, this advantage comes at the cost of an accuracy limited by the granularity of the non-parametric model. While hierarchical methods help to decrease the complexity, the database size cannot be increased “ad-infinitum”. On the other hand, the accuracy of generative methods is limited only by the correctness of the model, at the cost of the need of a good initialization of the model given the locality of such approaches.

A hybrid approach seems promising to us. Discriminative approaches such as the one described in this section provide a good approximation of the hand pose with a small computational cost. Generative approaches can take advantage of such initialization, optimizing the hand pose locally. We believe that generative methods like the one described in Chapter 5 in the context of robot grasp execution can improve the accuracy of our hand pose estimation system with a manageable computational overhead.

2.5.3 Further integration with context information

So far we have considered the estimation of the hand pose given the hand appearance and the object occlusions. However, there is a number of sources of information that have been ignored in our system, Figure 2.26. First, the pose of the arm is highly correlated with the wrist orientation given the limited range of movements of the wrist joints. Second, while object occlusions provide some information about the hand pose, such information would be richer if the object is fully recognized and located. Moreover, in previous work we have shown that there is mutual benefit in the integration of different perception elements in the context of grasping [7], so arm pose estimation and object recognition systems would benefit from estimations of the hand pose. For all this we consider promising the integration of such systems with our hand pose estimation module.

2.5.4 Hand Model Registration

In this chapter we have used a single appearance model for the human hand. However, the shape of the human hand varies across persons not only in scale, but in inter-finger length ratio and finger-palm size ratio, for example. Not including these variations in our synthetic model limits the maximum accuracy of our system.

One possibility to cope with this problem is to adapt our generic model to specific subjects. Shape registration methods such as [156] can adjust the pose and local scale of a given model to make it match an observed shape. This could be performed as an initial phase of our grasping by demonstration procedure each time a new person wants to demonstrate an action to the robot.

Allowing shape deformations during a demonstration could help also coping with shape variations due to contact with objects. However, shape registration takes several seconds per frame, so nowadays it is not possible to continuously adjust the shape of a model in real-time.

Chapter 3

Modeling grasping actions

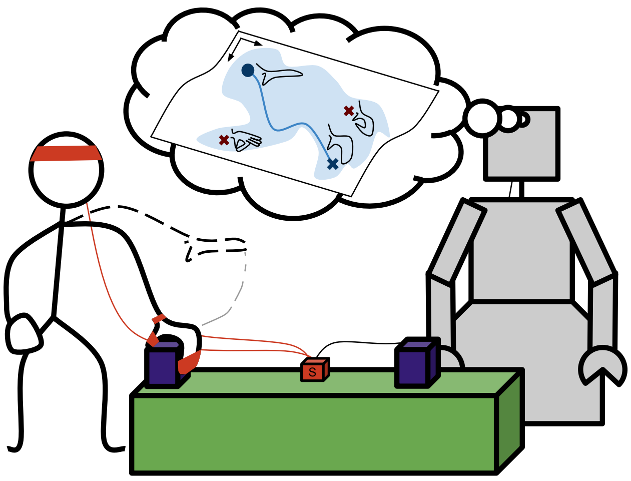


Figure 3.1: In this figure the robot is learning a representation of the grasping action (blue path) and of the grasping space of actions (blue blob) being executed by human teachers. The grasping perception occurs in a “Sensors on Teacher” (Figure 1.2) framework. The robot maps this demonstration to a lower dimensional space, together with other demonstrations of different grasps and different teachers, in order to get a generic representation of the grasp

Depending on which aspects of grasping actions are observed, grasping objects can be seen as a trivial task, a difficult one or just something less complex than it seems. For example, if a person with no physical disabilities is asked to rate the difficulty of grasping objects in his daily life, he would probably answer that grasping objects is trivial. The apparent simplicity of grasping for human subjects resides in the low requirements it imposes in terms of concentration and intellectual or physical effort. However, when the task of grasping is observed at a kinematic or muscular level, the task suddenly becomes incredibly complex. Several muscle activation signals have to be sent and coordinated by the central nervous system to actuate the muscles according to the task requirements. Those activation patterns not only depend on a pre-computed plan for the grasp, but can also be adapted online to new stimuli in the form of proprioception and external sensing. Persons with disabilities in the control of these signals (such as Parkinson disease patients) or with limited perception (such as people using prosthesis without tactile feedback) would say that grasping tasks are difficult, because the underlying control of the grasping action is not fully functional for them.

In the context of robotics the difficulty of the underlying control of grasping actions cannot be obviated. However, we believe that its complexity is lower than it seems. The difficulty of such task comes principally from the large number of variables to be coordinated and controlled. Nevertheless, those variables are not independent, and therefore the correlations between them can be exploited to simplify control. Apart from being intertwined with each other, the variables that define hand poses in a grasping action are also highly auto-correlated (in time). It becomes clear that the information required to described a grasping action is much more compact than it seemed initially, and its control is simpler.

In this chapter we explore the existing correlations in human hand poses when performing grasps. The study of those correlations results in compact models of human grasps that can provide us insight into grasp similarity, robotic hand anthropomorphism and how the hand pose changes in time during a grasp. Moreover, we provide detailed information about the methodology used to obtain such models, its advantages and how to interpret the obtained models.

3.1 Introduction

Human actions and their inherent complexity are not new research topics; as early as the 30s, the neurophysiologist Nikolai Bernstein posed the "degrees of freedom" problem. He foresaw in [34] a problem in the integration of proprioception with the non-univocal relationship between muscle contraction and kinematic output. Correlations between different components of human movements were found. As an example, in a hammering action the direction of the elbow is highly correlated with the velocity and angle of the hammer (among other characteristics). He also described cyclical actions such as walking in terms of Fourier series. From this study he concluded that few components determine the outcome of an action and that actions should be considered as a whole instead of series in time of independent motor commands. Later, in 1940, he used the nowadays widespread term "synergy" as the integration and coordination of large numbers of actors (in different levels, e.g. muscles or limbs) in locomotive actions like walking, [33].

The concept of synergies has survived the pass of time and is still heavily used today. Its use is common both in neurophysiology and in robotics with different wordings, being synergies and motor primitives the two mostly used ones. Both of them relate to a compact representation of actions that simplifies control. Pribham et al. [167] showed that there exists groups of neurons narrowly tuned to frequencies the forelimb of a cat is moved with. Similarly, rhythmic movements have been learnt and controlled through the usage of non-linear oscillators in the robotics world, [111]. In that work, basic actions (movement primitives) are represented compactly by a set of attractors (either points for discrete movements or cycles) which determine a control law. These control laws are learnt by imitation and can be adapted to conform in terms of reach points, frequency, etc. These motor primitives can also be learnt through reinforcement learning, [159].

While motor primitives usually refer to compact representation of motor control both in robotics [111] and in neuroscience [202], synergy applies more widely to low dimensional representations of movements. However, those terms are sometimes used indistinctly, [87]. Grinyagin et al. [98] classifies synergies into three types. First, static postural synergies, which refer to the correlation between single kinematic poses, e.g. [175]. Second, kinematic synergies dig into the correlation in time of postures during an action, e.g. [136]. Finally, muscle synergies concern the covariation of lower level representations of movement such as electromyographic activity, [212, 203].

There is evidence that muscle synergies, interpreted as time patterns in muscles coactivation, exist both in humans and animals. These synergies differ from the general concept of motor primitive in the sense that they can be activated simultaneously. Each of these synergies has a functional interpretation in terms of specific forces at the endpoint of the hindlimb of a cat [203]. This finding suggest that such synergies could be directly related to the functional control of task-level variables. A similar concept has been used in robotics, where task-space control is learned through the usage of manifolds in joint-space, [39].

Focusing on the particular problem that concerns this thesis, there is a substantial amount of research targeted towards synergies in human and robot grasping. The contrast between the complexity of grasping and the lack of effort it requires from humans attracted the attention of Friedman et al., [89]. Their study, performed from the point of view of neurophysiology, focuses on contact position, velocities and forces exerted while grasping. The finger placement for different grasps was compared among different persons, concluding that the variability is high, and that it propagates to other variables studied in the experiment. However, the correlation between rotational stiffness and the opposition axis of the grasp was detected. In [149], correlations between wrist and finger movements were modeled, validated and applied to solve control of redundant degrees of freedom.

There exists different algorithms that can be applied to learn the correlations we are looking for. In the machine learning community they are referred to as Dimensionality Reduction techniques. Such algorithms are instances of unsupervised learning aiming to recover the intrinsic representation of the data from the observed instances. Conceptually, the intrinsic representation is the space where each effective degree of freedom in the data is represented by a single dimension in an orthogonal space. The intrinsic representation retains all the variance in the data with the lowest possible number of dimensions. A popular method that strives to recover such representation is Principal Component Analysis (PCA, [115]). For a given dimensionality, PCA computes the intrinsic representation which maximizes the retained variance under an important assumption: the intrinsic representation is a linear transformation of the original dimensionality of the data.

PCA has been the main tool used to model possible pose correlations in grasping. One of the earliest examples corresponds to Santello, [175]. The authors showed that a great part of the variations of grasping hand poses (80% of the data *variance*) can be modeled with a two-dimensional manifold.

While Santello showed the correlation of joints for different static hand poses, later research focused on the temporal correlation (i.e. kinematic synergies) of the hand pose while executing specific grasps. In [98], multiple executions of precision grips are analyzed with PCA to conclude that a one-dimensional synergy can explain more than 97% of the data variance. The data from six different subjects and three different grasping conditions was analyzed separately, generating a different one-dimensional manifold for each of these series of twenty trials. Mason et al. [136] studied the correlation in the position of different parts of the hand for specific grasps applied to different objects, using again similar techniques to Santello. He concluded that for each subject and grasp, more than 96% of the variance could be explained by a one-dimensional manifold.

The concept of synergies have also had an impact in research on robotic control. Ciocarlie et al. introduced in [56] the concept of eigengrasps based on the grasp synergies defined in [175]. In such system, grasps are planned in terms of eigengrasps instead of manipulator degrees of freedom, making the optimization of the hand pose more computationally tractable. Moreover, eigengrasps unify the dimensionality of different manipulators in a similar way to the concept of Virtual Fingers, [20] (more information about virtual fingers in Chapter 4). Eigengrasps were also used in [96] to control the 12 degrees-of-freedom of a robotic hand. In [207] the grasping control of a 17-dimensional hand was performed by moving around a 2-dimensional manifold extracted with Isomap. Another area of robotics influenced by the concept of synergies is hand design, as we will see in Section 3.6.4.

3.1.1 Modeling Challenges

We found the concept of synergies appealing for the creation of models for our grasping-by-demonstration system. To that end, we would like to employ unsupervised learning algorithms in order to find efficient representations of dynamic grasping data for which the complexity of the modeling task can be significantly reduced. However, our problem poses some requirements on the synergies that were not met by any of the examples described above.

First, we want to gain insight into user-independent synergies. The reason for this is that our modules should be robust to input obtained from different users. The creation of user-independent synergies require data from different users to be integrated in a common space, as in [175]. The rest of the systems above use data from different users to create user specific synergies.

Second, we want all grasp types to be modelled on the same space, i.e. we want to create a common space X where all the grasps g_i have a representation $\{x_t^i\}$. This is shown in Figure 3.2; every node in the left-most module represents a grasp type, and all of them describe a path (left bottom) on a common two-dimensional space X. This allows us to simply map any new high dimensional pose y_t or sequences of poses $\{y_t\}$ into X

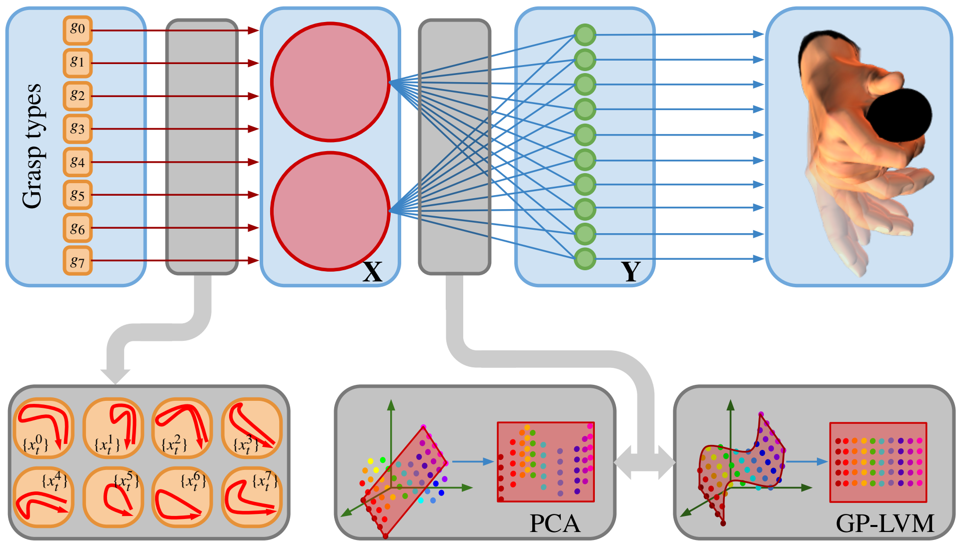


Figure 3.2: The figure above shows the schematic description of the system for modeling grasping actions that we evaluate and analyse in this chapter. We model different grasp types g_i (top left box) as time series of poses $\{x_t^i\}$ in low dimensional space **X**. We can see eight examples of those models in the lower left box. The low dimensional (two-dimensional in this example) space **X** is extracted from the high dimensional space **Y** using unsupervised dimensionality reduction methods, such as PCA (bottom-middle) and GP-LVM (bottom-right). The mapping between **X** and **Y** depends on the dimensionality reduction method used; PCA forces a linear mapping (i.e. assumes data distributed in hyperplanes) while GP-LVM allow non-linearities with its mapping based on Gaussian Processes. The hand poses (left box) are fully described by the space **Y**.

and perform prediction (which pose x_{t+1} is most likely) or classification (which grasp g_i is most similar to the sequence $\{x_t\}$) in the low dimensional space X. Examples of such common spaces for different action synergies can be found in nature. For example, varying the current strength of a single signal (one-dimensional space X) stimulating the midbrain of the salamander makes it to switch between stepping and swimming, [47]. Santello et al. also modelled a common synergy space for the static grasp poses. However, Mason et al. [136] grouped data from different objects onto the same manifold, but did not group different grasps. Grinyagin et al. also created a different synergy space for each grasp.

Third, we want to consider the grasping sequence as a whole, instead of considering only the hand pose while grabbing the object (kinematic synergies instead of static pose synergies, [98]). According to Bernstein [34], the problem of degrees of freedom is inseparable of the concept of action as a whole; temporal correlation is important. In our system, we want to understand better the evolution in time of hand posture while grasping. Observing such evolution in time can also provide useful information for robot executions of the grasps. Therefore, we should create synergies which include different time-steps of human grasps. While this was performed in the studies by Grinyagin [98] and Mason [136], it was not studied by Santello [175].

Integrating time-varying hand poses of different users performing different grasps into a common low dimensional space is a challenging task. While the hand poses from a single subject executing a particular grasp can be embedded into one-dimensional manifolds with high fidelity (more than 96% of the variance, [98, 136]), dealing with multiple users and grasps already requires two dimensions in order to explain 80% of the variance, [175]. Including time-varying poses into a multi-user, multi-grasp manifold will require either a higher dimensionality or to reconsider the assumptions of the techniques used.

PCA methodology can be too restrictive, as we can see in the bottom part of Figure 3.2. The colored circles represent “high dimensional” hand poses. Since they lie on a corrugated plane, PCA cannot recover the two-dimensional structure of the data properly, missing a substantial amount of variance in the direction perpendicular to the planar manifold. Methods with a more general mapping that allow non-linearities, such as GPLVM, can potentially recover properly the structure of the data.

These limitations of PCA were not made explicit in neither of the studies presented in [175, 98, 136]. Even studies comparing different methods for computing synergies like [206] contemplate only linear methods. Studying the limitations of the methods traditionally used in this field led us to compare a non-linear, generative method (Gaussian Process Latent Variable Models, GP-LVM) to the omnipresent PCA.

We present a thorough evaluation of the implications made by the application of different algorithms. Further, we show that dimensionality reduction algorithms are far from being black-box algorithms and that careful consideration of the data can significantly improve the results. We exploit this and show how prior knowledge can be incorporated into the framework in a principled manner.

The chapter is structured as follows. In Section 3.2 we present the mathematical tools to derive the synergic representations using PCA and GP-LVM, whereas in the following Section 3.3 a more intuitive overview of dimensionality reduction is given. We present how the data was generated in Section 3.4 and the resulting low dimensional spaces are described in Section 3.5. The evaluation of the synergies comprises the study of the reconstruction error generated by each of the approaches in Section 3.6.1, the study of the semantics of the manifold in Section 3.6.2, and an example of usage of synergies in the context of prosthesis hand design in Section 3.6.4. Finally a summary, together with future work, is presented in Section 3.7.

3.2 Methods for Extracting Postural Synergies

As stated in the introduction, application of postural synergies is fundamentally entwined with techniques for dimensionality reduction where, given a large set of data, the latter is applied in order to extract the synergic representation. In order to avoid confusion we will formalize some of the fundamental notions discussed throughout this chapter. In specific we will refer to any type of observations or measurements as data. The parametrization of the data types presented here will also be referred as the representation of the data. We will deal mostly with two specific representations of the data: first, the observed representation which is the "raw" form of the data in the parametrisation it has been given to us ($\mathbf{Y}$ in Figure 3.2), and second the intrinsic representation is the parametrisation of the data such that its aligned with its underlying generating parameters ($\mathbf{X}$ in Figure 3.2).

Different dimensionality reduction methods make different assumptions about the data. Therefore, the application of different techniques will often result in different representations $\mathbf{X}$. In this section we will motivate and formulate the problem of dimensionality reduction. We will then introduce the two different approaches evaluated in this chapter.

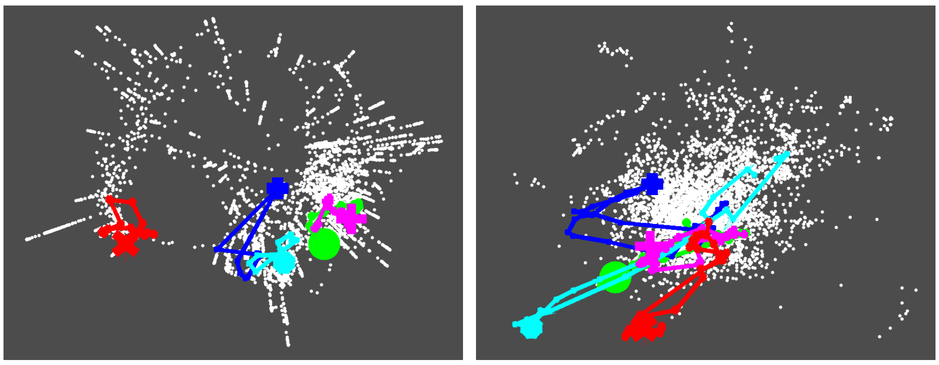
In many scenarios, the data is observed in a representation that is significantly different from the intrinsic representation of the data. In specific, this often implies that the observed representation is an "over" representation of the data in terms of degrees-of-freedom because the data actually lies on or close to a lower dimensional manifold in the observed representation. The task of dimensionality reduction is to, given raw data, recover the intrinsic representation. The problem is formalized as follows. Given a set of data $\mathbf{Y} = [\mathbf{y}_1, \dots, \mathbf{y}_N]$ where $\mathbf{y}_i \in \Re^D$ we assume this to have been generated from a intrinsic representation $\mathbf{X} = [\mathbf{x}_1, \dots, \mathbf{x}_N]$, $\mathbf{x}_i \in \Re^q$ through the generative mapping f,

$$\mathbf{y}_i = f(\mathbf{x}_i). \tag{3.1}$$

Further, we will assume the observed representation to be an over parametrisation implying that $q < D$. The objective of dimensionality reduction is to recover $\mathbf{X}$ from $\mathbf{Y}$.

The problem is severely ill-constrained as an infinite combination of input representations $\mathbf{X}$ and mappings f could have generated $\mathbf{Y}$. Different algorithms make different assumptions in order to proceed. There are two main branches of work in dimensionality reduction, *spectral* and *generative*. Spectral approaches assume the generative mapping f to have a smooth inverse. This is different compared to the generative class of models which directly tries to model the generative mapping. The spectral assumption is stronger and does therefore constrain the solution space further compared to the generative. This implies that while the generative models are applicable to a larger range of data, recovering the solution might be pose significant challenge. There are both linear and non-linear formulations of the methods.

Apart from the methods that will be described in depth in this chapter (PCA and GP-LVM), we initially considered two more methods: Isomap and Locally Linear Embeddings (LLE). Isomap, [198], estimates the intrinsic representation based on the geodesic pairwise distance between the points. By using the geodesic instead of Euclidean distance, Isomap manages to unravel highly non-linear manifolds. However, the presence of noise can create "shortcuts" in the manifold, producing erroneous geodesic distances. LLE, [174], replaces the PCA assumption of global linearity with local linearity: points are described as linear combinations of neighboring points. The dimensionality is reduced while preserving this locally-linear geometry. Both of these methods are popular non-linear dimensionality methods. However, both of them are based on local distance measurements, which is very sensitive to noise. After inspecting initial results (Figure 3.3), we concluded that these techniques were not applicable to our problem. Both in LLE and Isomap the same grasp



(a) Grasp 1 modeled with Isomap (b) Grasp 1 modeled with LLE.

Figure 3.3: Mapping of the hand poses of 5 subjects (different colors) performing a specific grasp. The mapping diverges too much across users; therefore we cannot extract a single synergy valid for multi-user data with these approaches.

executed by different subjects was too different, making impossible the creation of a single model for a grasp (Figure 3.3).

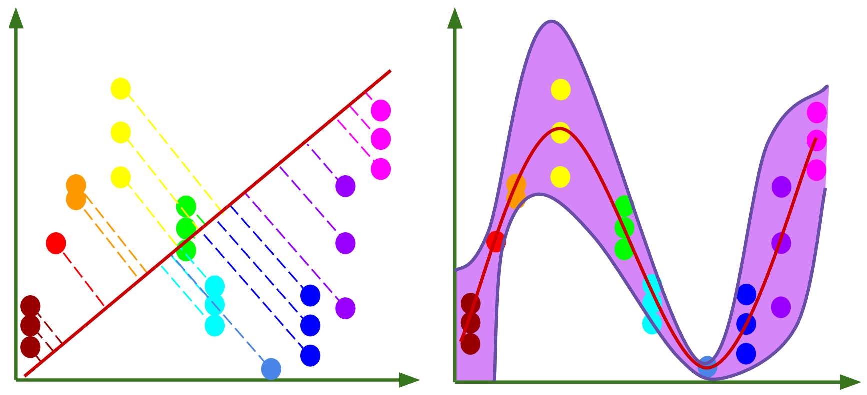
From now on, we will focus on the PCA and GP-LVM approaches. PCA is a spectral linear model while the GP-LVM is generative and capable of modeling a non-linear generative mapping. Our motivation for evaluating the performance of these two methods stems from the fact that PCA has been the dominant algorithm for extracting postural synergies while the GP-LVM is one of the most recently proposed and flexible algorithms in the area of dimensionality reduction, used in recent systems for people tracking [209]. The reminder of this Section will describe the different methods and further motivate our choice.

3.2.1 Principle Component Analysis

Principle Component Analysis (PCA) is a frequently applied method for dimensionality reduction over a large range of application fields. The objective of the algorithm is to find a low dimensional representation which minimizes the difference between the real covariance matrix and its approximation. Formally, we look for a low-rank approximation of the covariance matrix of the data under the Frobenius norm, minimizing the following cost:

$$\mathrm{E}(\mathbf{C}) = \|\mathbf{Y}^{\mathrm{T}}\mathbf{Y} - \mathbf{C}\|, \tag{3.2}$$

where $\mathbf{Y}$ is the centered observed representation of the data and $\mathbf{C}$ the approximation to the covariance matrix. The optimal solution to Eq. 3.2 can be found in close form through the eigendecomposition of the covariance matrix in the observed space,



(a) Linear mapping for PCA (b) GP mapping for GP-LVM

Figure 3.4: Mappings from two-dimensional data to one-dimensional manifolds, performed by PCA (left) and GP-LVM (right). In PCA, the result of the mapping is a line. In GP-LVM, the result of the mapping is a Gaussian Process, which has a mean (central red line) and covariance (shaded purple area). The covariance represent the uncertainty of the process in a particular point.

$$\mathbf{C} = \sum_{i=1}^{D} \lambda_i \mathbf{v}\mathbf{v}^{\mathrm{T}}, \tag{3.3}$$

where λ_i and $\mathbf{v}_i$ are the $i : th$ eigenvalue and vector of the eigendecomposition of the covariance matrix in the observed space. Through the definition of the eigendecomposition $\mathbf{v}_i$ specifies an orthonormal basis. Therefore, the "mass" provided by each component is proportional to the corresponding eigenvalue. The best rank k approximation of the covariance matrix is then given by,

$$\hat{\mathbf{C}}_k = \operatorname{argmin}_{\mathbf{C}} \mathrm{E}(\mathbf{C}) = \sum_{i=1}^{k} \lambda_i \mathbf{v}\mathbf{v}^{\mathrm{T}} \tag{3.4}$$
$$\lambda_i \geq \lambda_j, \quad i < j.$$

Consequently, the generative mapping takes the following form $f(\mathbf{X}) = \mathbf{V}_{1\rightarrow k}^{\mathrm{T}}\mathbf{X}$ and that the intrinsic representation found by PCA is $\mathbf{X} = \mathbf{V}_{1\rightarrow k}\mathbf{Y}$. This shows the fundamental assumption of PCA, that the generative mapping takes linear form 3.4. The major benefit of the algorithm is that it is robust as it relies on global statistics in the data. Further, if it can be assumed that the noise in the data is of low variance and that the intrinsic signal occupies a linear subspace in the observed representation then it will recover the correct space. However, these are strong assumptions that cannot be made reliably for many types of data as often a significant portion of the variance corresponds to noise and/or that the correlations in the data are non-linear.

3.2.2 GP-LVM

The Gaussian Latent Variable Model (GP-LVM) [128] is a generative dimensionality reduction model. As such it aims to directly model the generative mapping from the intrinsic $\mathbf{X}$ to the observed $\mathbf{Y}$ representation of the data. Intuitively, GP-LVM optimizes the internal representation $\mathbf{X}$ and the parameters of a family of mappings f so that the probability of the observed data $\mathbf{Y}$ is maximized. One of the main differences between GP-LVM and PCA is that the mappings in GP-LVM are more general (capable of reproducing any function) and probabilistic: instead of being a deterministic function, they have a mean and a covariance (Figure 3.4).

Different generative algorithms proceed in different manners to recover the intrinsic representation $\mathbf{X}$. In the GP-LVM framework the generative mapping f is modeled using a Gaussian Process ($\mathcal{GP}$) prior [169]. A $\mathcal{GP}$ is a set of random variables, any subset of which follows a joint Gaussian distribution. Defining the process is a mean function $\mu(\cdot)$ and a co-variance function $k(\cdot,\cdot)$ specified by a set of parameters θ referred to as the hyper-parameters of the $\mathcal{GP}$. Introducing a $\mathcal{GP}$-prior we can integrate over the generative mapping which leads to the marginal likelihood of the data,

$$p(\mathbf{y}|\mathbf{X},\sigma) = \int p(\mathbf{y}|\mathbf{f},\sigma)p(\mathbf{f}|\mathbf{X})\mathrm{d}f. \tag{3.5}$$

where $p(\mathbf{y}|\mathbf{f},\sigma)$ takes the form of a product of Gaussians, since we assume conditional independence of the points y_i given the mapping f and Gaussian noise $\epsilon \sim \mathcal{N}(\mathbf{0},\sigma^{-2}\mathbf{I})$ with variance σ:

$$p(\mathbf{Y}|\mathbf{f}) = \prod_i \mathcal{N}(\mathbf{y}_i|f(\mathbf{x}_i),\sigma^{-2}), \tag{3.6}$$

The previous formulation of the marginal likelihood is invariant to the scale of the latent space. In order to avoid this, an uninformative prior $p(\mathbf{X})$ is introduced in the formulation.

The GP-LVM proceeds by finding the latent locations $\mathbf{X}$ and the hyper-parameters θ that maximize the marginal likelihood with respect to the observed data. Modeling each observed dimension with by an independent $\mathcal{GP}$ leads to the following marginal likelihood,

$$P(\mathbf{Y}|\mathbf{X},\theta) = \prod_{j=1}^{D} \frac{1}{(2\pi)^{\frac{N}{2}}|\mathbf{K}|^{\frac{1}{2}}} e^{-\frac{1}{2}y_j^T\mathbf{K}^{-1}y_j} \tag{3.7}$$

as a product of D independent Gaussian processes.

3.2.2.1 Covariance Functions

The covariance matrix $\mathbf{K}$ in Eq. (3.7) is determined by the kernel or covariance function k:

$$\mathbf{K}_{i,j} = k(\mathbf{x}_i,\mathbf{x}_j) \tag{3.8}$$

The choice of covariance function encodes our preference about which possible correlations can exist between the data points, and which type of correlation is more important. This is an advantage of the method, since it allows adaptation to the specific needs of the task and the dataset at hand. The kernel function needs to generate a valid covariance matrix, i.e. a positive semidefinite kernel matrix.

A linear combination of valid kernel functions will also generate a valid covariance matrix. To that end we will use a combination of a Radial Basis Function (RBF), bias and white noise kernel. The RBF kernel is defined as follows:

$$k(\mathbf{x}_i, \mathbf{x}_j) = \sigma_r \, e^{-\frac{\gamma}{2}(\mathbf{x}_i-\mathbf{x}_j)^T(\mathbf{x}_i-\mathbf{x}_j)} \tag{3.9}$$

where σ_r defines the output variance and the inverse kernel width γ controls the smoothness of the function. A large inverse width will reduce the penalisation with respect to the smoothness of the generative mapping making quickly varying functions more prominent in the prior. Clearly this will constrain the solution less allowing a better fit to the data, however, with that comes the over-fitting of the data. A constant offset is included in the covariance function by introducing a bias kernel,

$$k(\mathbf{x}_i, \mathbf{x}_j) = \sigma_b, \tag{3.10}$$

which will account for translations in the data. The Kronecker's delta function can be used to explain variance that cannot be correlated to other points in the data, i.e. white noise. The amplitude of the noise is controlled via σ_n,

$$k(\mathbf{x}_i, \mathbf{x}_j) = \sigma_n \delta_{ij} \tag{3.11}$$

In this chapter we will use a linear combination of the three above kernel functions to parametrise the covariance function that specifies the prior over the generative mapping. This results in the following kernel function,

$$k(\mathbf{x}_i, \mathbf{x}_j) = \sigma_r \, e^{-\frac{\gamma}{2}(\mathbf{x}_i-\mathbf{x}_j)^T(\mathbf{x}_i-\mathbf{x}_j)} + k(\mathbf{x}_i, \mathbf{x}_j) = \sigma_n \delta_{ij} + \sigma_b, \tag{3.12}$$

resulting in a model specified by four hyper-parameters, $\{\sigma_r, \gamma, \sigma_n, \sigma_b\}$.

3.2.2.2 Back Constraints

By using a smooth covariance function like the RBF kernel, we encode a preference towards smooth generative mappings in the GP prior. This implies that points close in the latent space will remain close in the observed space. The opposite is not guaranteed though; i.e. points close in the observed space remain close in the latent space. However, this can be incorporated into the model by representing the latent locations $\mathbf{x}_i$ in terms of a smooth parametric mapping g_j from the observed data $\mathbf{y}_i$. In specific we are going to use a mapping that is capable of modeling non-linear correlations by employing a regression model over a kernel induced feature space,

$$x_{ij} = g_j(\mathbf{y}_i, a) = \sum_{n=1}^{N} a_{jn} k_{bc}(\mathbf{y}_i, \mathbf{y}_n), \tag{3.13}$$

where k_{bc} will be referred to as the back constraint kernel. This means that the maximum likelihood solution of these parameters a rather than the latent locations are sought. This is referred to as a back-constrained GP-LVM [127]. In addition to constraining the latent location to preserve the local smoothness of the observed data, previously unseen data can be projected onto the latent space in an efficient manner by pushing them through this back-mapping. For many real life applications it is desirable to be project from data space to the latent space in an easy and fast way.

In the next section, we discuss the effect of the different assumptions and show what implications they have on the resulting representation. Further, even though being strictly unsupervised, we show how prior knowledge of the data can be included which can significantly improve results.

3.3 Interpreting the Models

The solution to both PCA and the GP-LVM is reached by minimising an associated energy function. In this section we begin by analyzing these different objective functions in order to provide further intuition about the data modeling. PCA aims to find a low-rank approximation that minimizes the Frobenius norm with respect to the covariance matrix in the observed space (Eq. (3.2)). Intuitively, this means that PCA searches for the hyperplane which minimizes the distance from itself to the high-dimensional datapoints (see Figure 3.4b). The objective of the GP-LVM is less obvious. The effect of maximising the marginal likelihood will be that of minimising the variance of the observation noise constrained by the $\mathcal{GP}$-prior (i.e., minimize the shaded area in Figure 3.4a)

The variance of a population is a measure of the "spread" of the data along a specific direction. Formally it is the expected value of the squared distance from the mean,

$$\text{Var}(\mathbf{x}) = \text{E}\left((\mathbf{x} - \bar{\mathbf{x}})^2\right), \tag{3.14}$$

where $\bar{\mathbf{x}}$ is the mean of the population. Similarly, the covariance between two variables is defined as,

$$\text{Cov}(\mathbf{x}, \mathbf{y}) = \text{E}\left((\mathbf{x} - \text{E}(\mathbf{x}))(\mathbf{y} - \text{E}(\mathbf{y}))\right), \tag{3.15}$$

indicating how the two variables change together. The covariance matrix K of a D dimensional multi-variate variable is a matrix whose elements k_{ij} contain the covariance between dimension i and j. Minimizing the distance between two matrices with respect to the Frobenius norm is equivalent to minimize the element-wise difference in a Euclidean manner. This means that each element in the matrix difference contributes equally to the distance. When applied to a covariance matrix, this implies that each dimension in the observed representation is given equal importance as they all contribute equally to the norm. Similarly, the noise in the observations are assumed to be distributed as a spherical Gaussian in the GP-LVM model. Therefore, when fitting the noise to the data each dimension in the data is given equal importance.

Central to the concept of variance is the notion of distance. Conceptually a distance is defined according to the representation of the data. As both PCA and GP-LVM minimize variance in some sense, the application of these algorithms depends on the representation of the observed data as it defines the distance it is computed from. The implications of this is that a dimension along which the data is further "spread" will give a larger contribution to the error function, which means that the algorithm will focus on modeling this dimension. This means that in order to achieve a proper representation we need that the distance in the observed representation corresponds well to the concept of "similarity" that we want to preserve. We can control the contribution of each dimension to the error function by pre-scaling the data. If prior knowledge is available, this can be exploited by transforming the data such that the L_2 distance better reflects our notion of similarity. Since we do not want to commit to any special purpose scaling for the grasping data we have collected, we scale each dimension to make it have the same variance; this process is called "whitening".

3.3.1 GP-LVM

The hyperparameters of the mapping and the latent locations of the GP-LVM model are obtained through an iterative procedure which tries to minimize a non-convex objective function. This means that we cannot guarantee to recover the optimal model which corresponds to the maximum marginal likelihood of the observed data. Therefore, once a solution is found it is important to inspect the embedding and the hyper-parameters, in order to evaluate the quality of the solution. In this section, we discuss how one can gain an intuition about the data and the solution from the resulting learned model.

3.3.2 Interpreting the Kernel Parameters

The covariance matrix K from Equation 3.7, also known as kernel matrix, defines the correlation between the point projections. This matrix is determined by the kernel function k, which has to be chosen. We use a kernel function (Section 3.2.2.1) composed by an Radial Basis Function (RBF), a bias term σ_b and a white noise term σ_n.

$$\mathbf{K}_{i,j} = k(\mathbf{x}_i, \mathbf{x}_j) = \sigma_r \, e^{-\frac{\gamma}{2}(\mathbf{x}_i - \mathbf{x}_j)^T(\mathbf{x}_i - \mathbf{x}_j)} + \sigma_b + \sigma_n \delta(i, j) \tag{3.16}$$

The parameters σ_r,γ,σ_b and σ_n are optimized by the GP-LVM optimization. So having a look at these parameters can provide some insight on how the model was able to cope with the presented data. Therefore, one can infer knowledge about the projection from the latent space to the high dimensional space.

The parameters σ are weighting the three kernel functions. Looking at the relative values of σ_r, σ_b, σ_n helps to determine which of the three kernel function is dominating the reconstruction. How much the RBF kernel affects to the total variance is governed by σ_r. The parameter σ_b, the bias term, adds an offset to all the elements in the kernel matrix. This means that it sets what is the minimum correlation between any pair of points. Finally, the term $\sigma_n \delta(i, j)$ corresponds to the noise term, and increases only the covariance value of every point with respect to itself, $\mathbf{K}_{i,i}$. This term is used to "explain away" points that are not supported by the actual model. A high value of σ_n then implies that a big part of the data is actually treated as noise instead of being supported by the model.

The RBF kernel has an additional parameter which is optimized, the inverse width γ. The RBF term depends on the ratio between the points distance and it's width, $\frac{1}{\gamma}$. It defines the size of the neighborhood which is used for reconstructing a single point in the high dimensional space. Therefore, when the optimized value of $\frac{1}{\gamma}$ is big (compared to the variance of the data in low dimensional space) we can infer that the points are highly correlated and many points are used for reconstructing a point. This results in trajectories that are smooth as the process is "averaging" over many points. On the other hand, if $\frac{1}{\gamma}$ is relatively small, the space will generalize poorly over new unseen data, since the space overfits single points in the training sets.

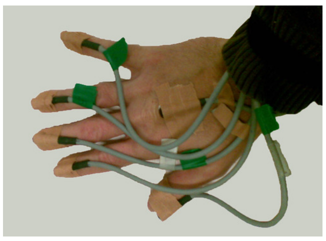
3.3.3 Back Constraints

In Sections 3.3.2 and 3.5.4, the projection from the latent space to the high dimensional space was discussed. The original GP-LVM does not provide a mapping from high dimensional space to low dimensional space. Additionally, there is no guarantee that a smooth trajectory of the high dimensional space will be mapped to a smooth one in the latent space. In order to overcome this limitations, the Back Constraint (BC) GP-LVM can be used. The latent positions are represented by a parametric mapping (Eq. (3.13)) from the high dimensional data space to latent space. As the name implies, this technique really generates a constraint as it encourages the relationship between the latent and the observed data to take the form of a bijection. This is a strong assumption which might significantly alter the solution. So one has to be careful on the influence of the additionally introduced constraints. The kernel matrix of Eq. (3.13) is controlling the mapping in a similar way as in Section 3.3.2. One difference is that the inverse width of the kernel is not optimized and has to be set, which gives some control over the latent trajectories smoothness. Having a very wide kernel means that the latent trajectories will be very smooth and this imposes a considerable constraint on the model which might reduce its ability to adapt to the data. On the other hand, if the kernel width is very narrow, the latent trajectories might become jagged. Additionally if one tries to project points that are far away (in terms of the kernel width) from the original dataset, the model might not have any evidence supporting this point to make a valid prediction. In that case the point will be projected to some point in the latent space where all unsupported points collapse to.

3.4 Data Description

The extraction of postural synergies by exploiting dimensionality reduction techniques is based on the fundamental assumption that we can acquire a data-set which "well" describes the problems state domain *i.e.* being sufficiently densely sampled. In this section we will describe the data-set we created for the work in this chapter. Such data-set is publicly available in `http://grasp.xief.net/`.

The data-set was generated from 5 subjects (3 male, 2 female). All subjects were right handed and no one reported any hand disabilities. The hand length is 185.2 ± 13.3mm and hand width is 81.1 ± 7.4mm. A Polhemus Liberty system with six magnetic sensors was used for recording the data. Each sensor provided its orientation and position with respect


(a) Five sensors are placed on the fingertips and one on the wrist.

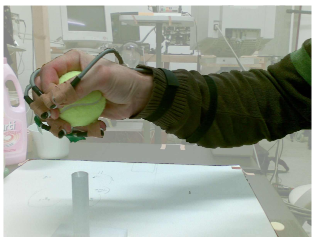

(b) Grasp posture for grasp 11.

Figure 3.5: Magnetic sensors setup

to a base point as a quaternion and a 3d vector (7 dimensions in total). The spatial and angular resolution of each sensor is 0.8 mm and 0.15 degrees respectively. One sensor was applied to each fingertip, positioned on the fingernail and one was placed on the dorsum of the hand. See Figure 3.5a for an image of the markers applied to the hand. The subjects were asked to perform 31 different grasp types [11] on an object typical for the specific grasp. If a subject had problems mimicking the grasp on the picture he was shown a picture and a demonstration of the grasp. To start they placed the hand in a flat posture on the table. Upon a starting signal they grasped an object with the desired grasp type, lifted the object (this moment is shown in Figure 3.5b), put it down again and retreated the hand to the starting position. The data recording started when the hand began to move and ended when the hand was put back to the initial position. Since we are interested in studying the intrinsic posture of the hand, we removed the global transformation from each grasp thereby representing each grasp in a common frame of reference.

Each grasp was discretized into 30 uniformly distributed time instances for which we record the joint positions. In summary this means that we acquired a database of 4650 poses consisting of five subjects performing 31 different grasps. Further, each subject was asked to perform each grasp twice, where we will use the first instance for testing and the second for training.

3.4.1 Task space vs Joint space

The sensors used in our system provide task-space data, i.e. position and orientation of the fingertips. This absolute data is converted into data relative to the wrist to focus on the inner hand kinematics instead of the transportation component of the movement. Some studies [175, 98] have preferred the usage of joint data, i.e. the joint angles of the hand.

We argue that task-space is preferable for our task. First, data in task-space is easier to compare, because, in joint-space data, proximal joints have a higher impact in the pose than distal joints. Second, task-space data is more “portable” between embodiments, because it directly encodes the relation of the hand with the object (the contact points and normals).

One disadvantage of task-space data in our case is that its dimensionality is higher. However, as we have argued before in relation with the rotation representation, it is preferable to start with a higher dimensional, well behaved space than with a problematic lower dimensional space.

3.4.2 Rotation Representation

Task-space data in our case involves three-dimensional position and orientation of the fingertips. This data will be interpreted as high dimensional vectors and compared in a Euclidean way both by PCA and GP-LVM. While the representation of positional data is straightforward, a representation of orientation which is "Euclidean-friendly" is less obvious. We explore different ways of representing orientation in the remaining of this subsection.

Euler angles are the most compact description of rotation in 3D space employing only three parameters. The main drawback of this method is that a smooth path in position-orientation space can correspond to a discontinuous path in euler angles. There are jumps in the data and additionally the method suffer the problem of a singularities at certain rotations angles (gimbal lock) [145]. In other words, the result of a small change in orientation might be a big change in those three angles. Therefore, comparing the euler angles as three-dimensional vectors does not reflect properly changes in orientation.

Quaternions use four parameters to define the orientation. Three parameters can be interpreted as the rotation axis vector and the last parameter is the amount of rotation around such axis. Besides some computationally advantages, this method is still very compact and yet it offers smooth transitions from one orientation to the other without singularities [145]. The main drawback of quaternions is that the Euclidean distance between them does not reflect their similarity. This becomes obvious when we consider quaternions the q and $-q$; the Euclidean distance between them is large, but they actually represent the same rotation (a rotation of w around axis x, y, z is the same as a rotation of $-w$ around the opposite axis).

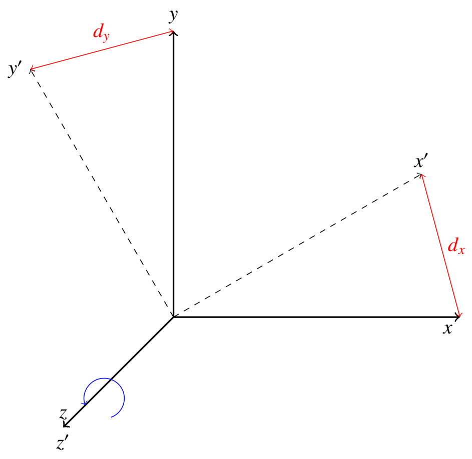
Rotation matrices use a 3×3 matrix to define the orientation. Those 9 parameters unambiguously define the orientation of an object at the cost of introducing many new dimensions to the dataset. The components of the rotation matrix can be seen as the three 3D vectors corresponding to the axis of the rotated system (Figure 3.6). GP-LVM and PCA would consider these matrices as flat, nine-dimensional vectors. The Euclidean distance between those nine-dimensional vectors is known as the Frobenius norm of the rotation matrix difference. This distance can be understood as the length of a vector composed by the distances between each of the components of the 3D axis (see Figure 3.6):

$$R = \begin{pmatrix} x_0 & y_0 & z_0 \\ x_1 & y_1 & z_1 \\ x_2 & y_2 & z_2 \end{pmatrix}, R' = \begin{pmatrix} x'_0 & y'_0 & z'_0 \\ x'_1 & y'_1 & z'_1 \\ x'_2 & y'_2 & z'_2 \end{pmatrix} \tag{3.17}$$

$$||R - R'|| = \sqrt{\sum_{i=0}^{2}(x_i - x'_i)^2 + \sum_{i=0}^{2}(y_i - y'_i)^2 + \sum_{i=0}^{2}(z_i - z'_i)^2} \tag{3.18}$$

$$= \sqrt{d_x^2 + d_y^2 + d_z^2} \tag{3.19}$$

Rotation matrices seem to be the most well-behaved representation for our rotations, since its Euclidean distance does not contain artifacts like ones mentioned for quaternions or euler angles. Their main drawback is its large dimensionality. However, it is preferable



$$R = \begin{pmatrix} x & y & z \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

$$R' = \begin{pmatrix} x' & y' & z' \end{pmatrix} = \begin{pmatrix} 0.866 & -0.5 & 0 \\ 0.5 & 0.866 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

$$||R - R'|| = \sqrt{d_x^2 + d_y^2 + d_z^2}$$

Figure 3.6: Euclidean distance between rotation matrices R and R' interpreted as nine-dimensional vectors. An orientation R is rotated 30° around z axis to obtain R'. The distance $||R - R'||$ between those orientations is the Euclidean norm of a vector composed by the distances d_x, d_y, d_z between each of the axis normal vectors.

to start the dimensionality reduction process from a well-behaved high dimensional space than from a lower dimensionality space with loops and bijections. Therefore we chose rotation matrices as the representation for the rotations in the system.

3.5 Synergic Representations

In Figure 3.2, a schematic figure of the evaluation framework is shown. We are particularly interested in evaluating the application of two different dimensionality reduction approaches for extracting postural synergies. This is shown by the gray modules in the right of the figure, where the middle approach is the traditional PCA approach which we compare to a back-constrained GP-LVM model (leftmost module). Such representation (the big red nodes in Figure 3.2 allow us to describe each pose as a vector in low dimensional space. However, as stated in Section 3.1.1, we are interested in representations that allow us to model a grasp from time-series data from multiple users. Such representation is build as an additional layer over the low dimensional representation. Each grasp type (orange nodes in Figure 3.2) is modeled with a Gaussian Mixture Regression (GMR) [48, 51] that predict over time in each synergic representation. The application of this dynamical model allows us to generate trajectories in the synergic representation of the data.

In the rest of this section we will present the representations extracted with PCA and GP-LVM, and how GMR is used to group together multi-subject grasps. The first contact with the synergic representation is done by plotting the projection of the training data into the low dimensional representation. By examining how the trials of different subjects performing the same grasp group together and how these types are distributed we will draw conclusions about how well the synergies represent the grasping data. Afterwards, we analyse the information that can be extracted from the kernel matrix, which will provide hints about the similarity between different grasps and what is the structure of the grasping actions.

3.5.1 Principal Component Analysis

PCA finds a rotated basis of the space in such a manner that subsequent components explain a decreasing amount of variance of the data, i.e. the first principal component (PC) explains most of the variance, the second PC explains most of the remaining variance and so on. This reasoning brings up the question of how many dimensions are enough for representing properly the observed data. The plot of the variance of the Principal Component in Figure 3.7 shows how by increasing the number of principal component the variance that is left unexplained decreases. The first component accounts for 59% of the variance, the second for 14% and the third for 5%. These numbers encouraged us to use 2D and 3D representations for our experiments, since the first three components explain close to 80% of the variance, and these representations are much easier to interprete visually than higher dimensional representations. Adding more dimensions does not contribute significantly to the variance, but increases the complexity of the representation.

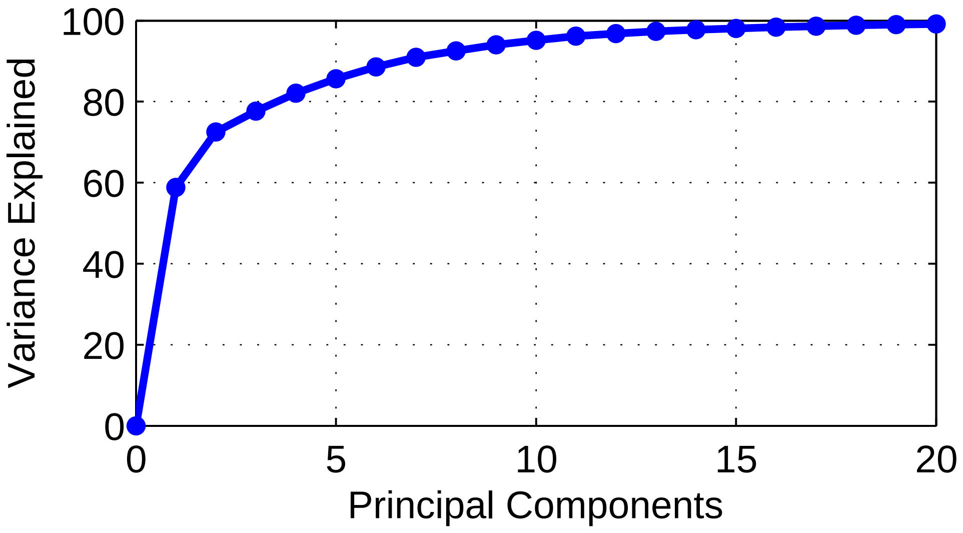
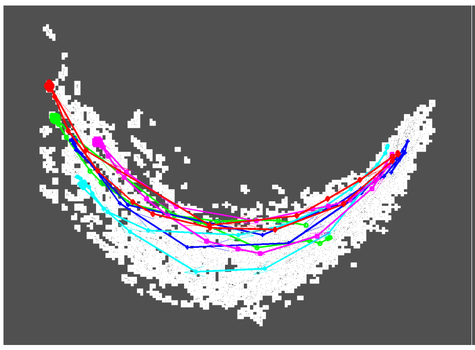
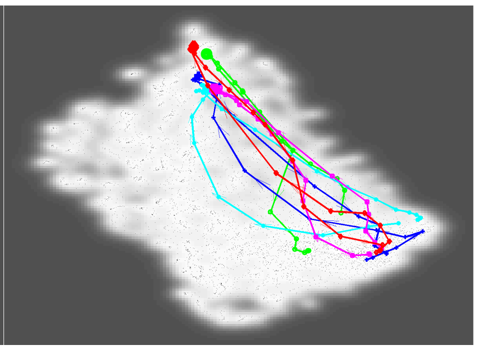
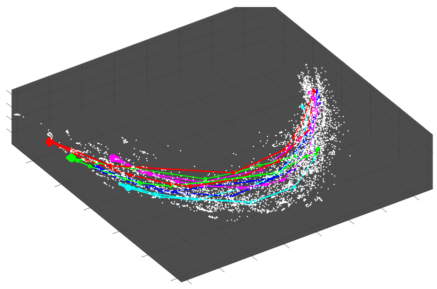


Figure 3.7: Variance of the data explained with increasing number of Principal Component. Only the first 20 components are shown, as the additional information transferred by the last ones is very small.


(a) PCA 2D space. The light dots represent datapoints and this gives an idea about the shape of the space.


(b) GP-LVM 2D space. The background image gives an idea about the shape of the space. The whiter the background, the lower variance has the model in that area (i.e. the reconstruction is more deterministic).


(c) PCA 3D space. The light dots represent datapoints and this gives an idea about the shape of the space.

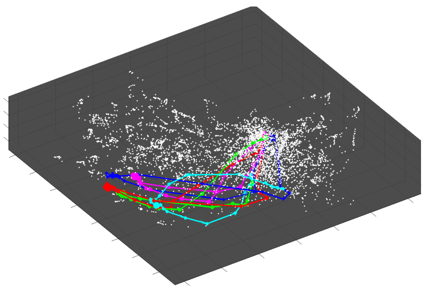

(d) GP-LVM 3D space. For simplicity, only datapoints (white dots) are represented, not the variance.

Figure 3.8: Comparison of the synergic representations of grasping data. In all figures the trajectories of the five subjects performing grasp number 4 are plotted in different colors.

PCA 2D (Figure 3.8a): To visualize the shape of the space the datapoints were plotted as white dots over a dark background. The point districution is a rather narrow ark, where all movements lie on. The initial starting posture is on the right side of the space, and when grasping the trajectory progresses leftwards. The final position of the grasp normally is the point farthest away from the starting region. The overall flexure of the fingers determine how far the trajectory moves away from the start point. The reason for this seems to be that the starting posture is a flat hand and therefore increasing the finger flexion increases the difference to the starting posture.

PCA 3D (Figure 3.8c): The 3D plots add the third PC to the data. As one can expect the variance is smaller than in the other two dimension. The arc structure is still very dominant in this dimension. The additional variance of the third PC is relatively small, nevertheless as shown in Section 3.6.2 this additional information can be used to better distinguish between grasp types.

3.5.2 GP-LVM

Each point in latent space is connected via a Gaussian Process mapping to a point in high dimensional space. It predicts the mean and the covariance of a point in high dimensional space given the latent location. The mean can be used directly as the reconstruction of a latent point in high dimensional space. The covariance, which is connected to the prediction, can be used to quantify the confidence the model has while generating the point in high dimensional space. A large covariance means that the model has a low confidence as it is poorly supported by data points. How fast the covariance increases while moving away from data points gives a hint on the ability of the model to generalize to previously unseen points. For the 2D model the background encodes exactly that covariance. In light areas the model has low covariance and therefore will deliver good predictions, whereas in the dark area there is high covariance (either due to observation inconsistencies or, most likely, lack of training points) and therefore the prediction gets uncertain. We plot the covariance only for the 2D case for simplicity's sake.

For the 3D case this visualization is not possible, even though the information is present. Therefore the plot is similar to the PCA visualization.

GP-LVM 2D (Figure 3.8b): Compared to the shape of PCA 2D (Figure 3.8a) the space covers a larger area. Since GP-LVM is a nonlinear method it was able to spread the grasp types better and therefore allows for a finer differentiation between them, as we will further explore in Section 3.6.2.

GP-LVM 3D (Figure 3.8d): Again the space is larger than the PCA counterpart (Figure 3.8c) which can be explained by the nonlinear capabilities of GP-LVM.

3.5.3 Summary

In the four extracted latent spaces the trajectories of different subjects performing the same grasp type showed a similar pattern. Nevertheless, for a given dimension GP-LVM is superior to PCA since it is manages to spread the points over a larger area. This allows for a finer differentiation between grasp types while keeping different user instances of the same grasp close to each other, and therefore for a richer description. Both methods are able to generalize between subjects. This means that given one grasp type, the trajectories of the subjects all show a similar pattern and move along similar paths in low dimensional space. That is the basis for the analysis in Section 3.6.2, where a model for each grasp type is generated using those five trajectories, defining in this way that particular grasp synergy. These models will also be compared in Section 3.6.2.

3.5.4 Interpreting the Kernel Matrix

The information described in the previous Section can be also visualized by the kernel matrix $\mathbf{K}_{i,i}$ directly. In Figure 3.9 we can see the kernel matrix corresponding to a 3D GP-LVM model, together with two magnifications of it. Black/purple means a low value (low correlation) while white/yellow means a high one. Overall the matrix has 4650x4650

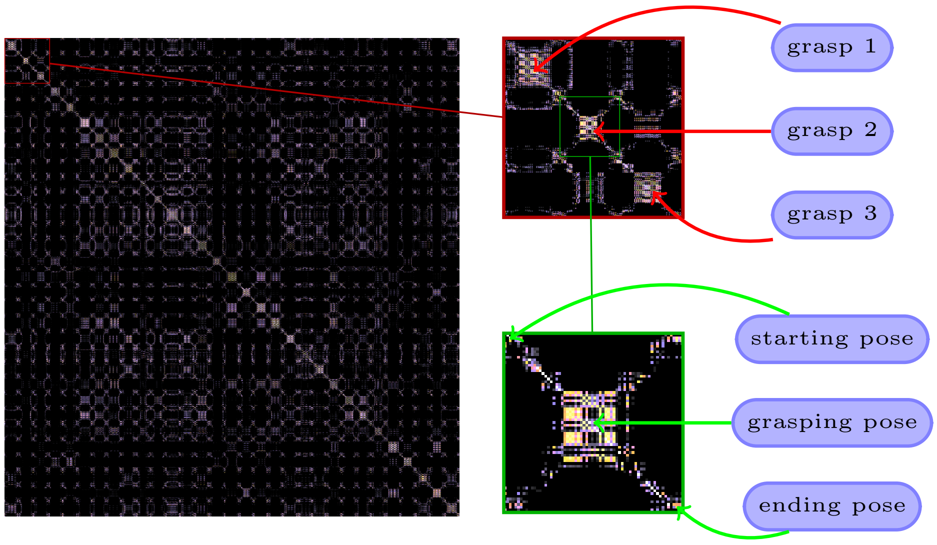


Figure 3.9: Kernel matrix. Black/purple represents low correlations, whereas white/yellow denotes highly correlated points. The left picture is the full kernel matrix, whereas on the right two magnifications of the matrix can be seen. The (i, j)-th pixel represents the correlation between the point i and j in latent space.

elements, where the (i, j)-th element represents the correlation between the points i and j in latent space.

The data is composed by 31 blocks corresponding to different grasp types, each of them divided into 30 timestep blocks, and finally each timestep block is divided into 5 subjects. We can see that the highest correlation occurs around the diagonal, where 31 blocks are present. This means that points are highly correlated with points of the same grasp (even from different users or time instances).

In the general view of Figure 3.9 we also see that there is a lattice made of small squares all around the matrix. As we see in the zoomed versions, these small squares correspond to the beginning and end of each grasp. The initial and final pose for each grasp are the same for every grasp type, so it is logical that there is a correlation between those points.

If we observe the upper right magnification, we can see that the central square (grasping pose) that relates grasp 2 and grasp 3 has higher values than the one for grasp 1 and grasp 3. This means that the correlation between grasps 2 and 3 is higher than between grasps 1 and 3. If we observe the pictures corresponding to the grasps from Figure 3.10, we can see that indeed the similarity between grasps 2 and 3 is higher than between grasps 1 and 3. These correlations can be observed in a better way if we crop the central part for each grasp, disregarding in this way the initial and final pose. Figure 3.11a shows the kernel matrix with points corresponding to frames 10 to 20 (among 30 frames in total of one

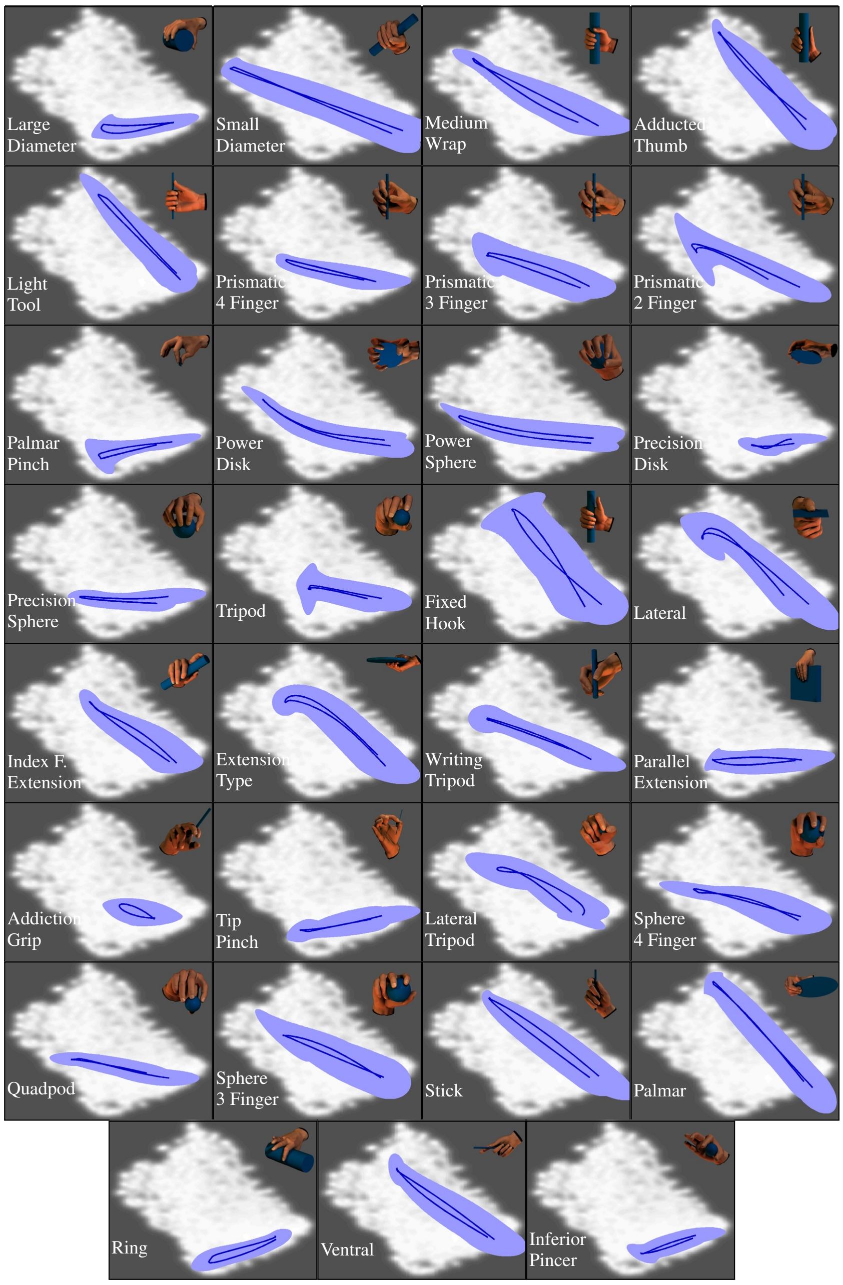
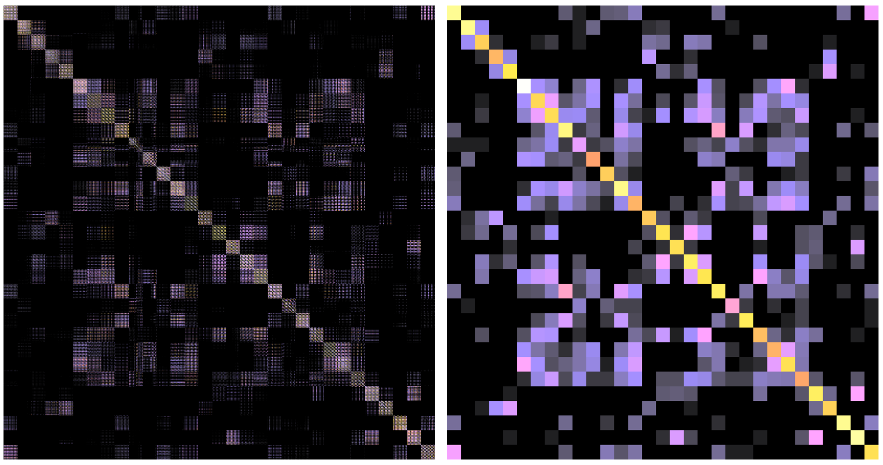


Figure 3.10: GMR regression on the 31 grasp movements of all subjects. The dark line indicates the mean trajectory and the light area correspond to the uncertainty. The grasps are sorted, so the first row contains grasps 1 to 4 and so on.

trial). Figure 3.11b shows the sum of covariance values for the 10 time frames and 5 users inside of each grasp type. The values in this matrix can be interpreted as the covariance of the classes in the data, and can be used as a simple way of determining which grasp is similar to which one. This matrix has size 31x31, as we have 31 different grasp types.



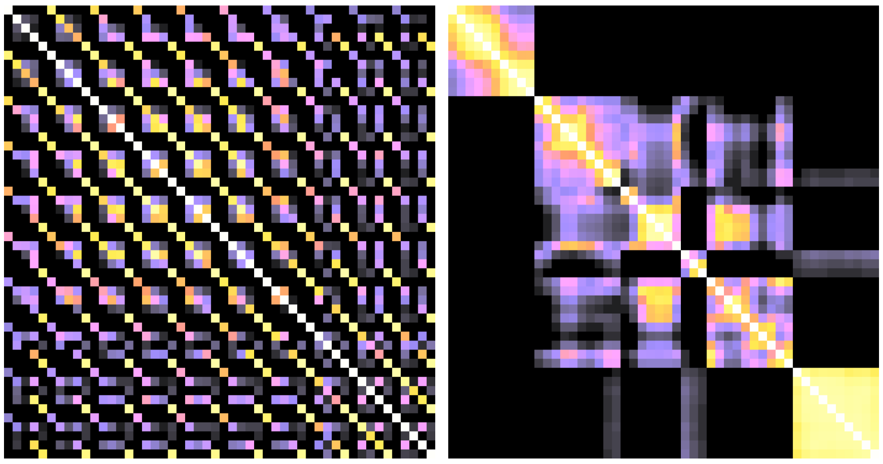
(a) Kernel matrix with cropped beginning and end. This means only the frames where the hand actually grasps the objects is presented in the figure.

(b) Mean of the left figure over subjects and frames. The matrix now has the size 31x31, which corresponds to the number of grasp types. So this figure shows which grasp types are similar.

Figure 3.11: Kernel matrix cropped.

In the lower right part of Figure 3.9 we see the magnification of the covariance values corresponding to points of a single grasp. Along the diagonal we can identify three areas; the first and last are replicated along every row and column in the general kernel matrix, while the central one is only replicated for the grasps which are indeed similar to each other. That means that the first and last sections correspond to poses largely similar among users and grasps, i.e. the approaching and retreat phase of the grasp.

Another detail of the kernel matrix worth of an explanation are the datapoints corresponding to grasp 21, as presented in Figure 3.11b. Grasp 21 has the lowest mean correlation with itself among all the grasps. The reason for this is a large variance among subjects, as can be seen in Figure 3.12a. It should be noted that the ordering in the left picture is that the first 5 pixels correspond to frame 1 of subjects 1-5, then frame 2 subjects 1-5 and so on. This creates this 5 pixel pattern. Once the matrix is rearranged to place the elements originating from the same user together (Figure 3.12b), it becomes visible that subject 1 and 5 are completely uncorrelated from the rest, and correlation between subjects 2, 3 and 4 is low. We can interpret this as a sign of extreme variance in the execution of grasp 21. Indeed, our experience recording the grasp sequences was that this grasp was executed among the subjects in very different ways. Grasp 21 is the “Cigarette” grasp, where an object is placed between the index and the ring finger. We only demanded that the object is



(a) Covariance grasp 21 (b) Covariance grasp 21 rearranged

Figure 3.12: Correlation of poses tagged as grasp 21. In the right the elements are rearranged to group elements from the same user. This shows that different subjects performed grasp 21 in a very different way.

grasped properly, but we did not specify how the remaining fingers have to be positioned. Some subjects kept the remaining fingers extended, while others flexed them.

It should be mentioned that this analysis is not only applicable to grasping data, but to any data with dynamical behavior through time and multiple classes.

3.5.5 Gaussian Mixture Regression of Grasps

As discussed before, the representation of each grasp in latent space should encompass temporal information (so that it is not just a point as in [175]) as well as multiple subject variance. We have used Gaussian Mixture Regression (GMR) [48, 51] for representing each grasp. We will briefly introduce this representation; more information can be found in [48, 51]. First the datapoints in latent space (two or three-dimensional data, see first column of Figure 3.13) are extended with a time dimension. Then this data (three dimensions) is fitted into a Gaussian Mixture Model (GMM, second column of Figure 3.13) by an expectation-maximization procedure initialized with K-means. Empirically, we found that using more than 3 Gaussians did not improve the quality of the fitting. Based on that mixture of gaussians a hand posture is inferred for each time step by using GMR. This creates a continuous path through the latent space that describes the grasp (third column of Figure 3.13). That path has a mean and a variance. The paths corresponding to each of the 31 grasps can be found in Figure 3.10.

The GMM/GMR representation of the grasps is a powerful tool that can be used for several purposes. One is the generation of new actions under some constraints, [51]. In our case, this could help to generate an action composed of two grasps without coming back to

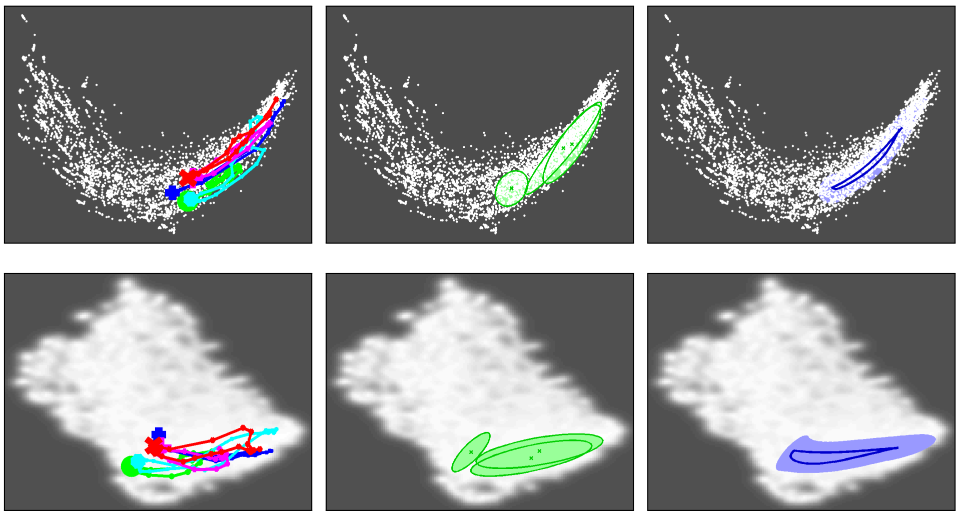


Figure 3.13: From top to bottom: PCA, GP-LVM. From left to right: projection of grasp number 1 into latent space, GMM fitting, GMR regression. The other grasp types show similar patterns.

the rest position between them. The second grasp can be constrained to start in a specific pose or after a specific time frame of the first grasp. The GMR can be optimized taking into account that constraint, providing in that way a smooth transition between those grasps.

3.6 Evaluation

In this Section we will evaluate the grasp synergies proposed by GP-LVM, and compare them with the ones extracted using PCA. We evaluate in Section 3.6.1 how well we can reconstruct poses that have been mapped to the low-dimensional space. Second, Section 3.6.2 evaluates the semantics of the synergies: how compact are the models of each grasp, how well can we discriminate between them and how can we use the similarity between grasps to construct a data-driven grasp taxonomy. Finally, a use case of the synergies presented in this chapter is shown in Section 3.6.4; we present how prosthesis hand design (and therefore robot hand design) can benefit from the grasping manifold we have built.

3.6.1 Evaluation of the Reconstruction Error

One important characteristic of the extracted synergies from a dataset is how accurately they represent the observed space of the data. This leads us to evaluate the quality of the learned mapping in terms of the reconstruction error. The reconstruction error shows how much the mapping connecting the observed and the latent representation distorts the data. It takes a point from the high dimensional space and pushes that point through the latent space and back to the original space. The difference between those two points is then calculated. That is performed for the both the training points, which tests how well the model adapts to those points, and the test set, where it allows us to access the performance for points which are new, but similar to some extent to the training data. Since no information about the classes of the grasp types is processed, it does not test the generalization ability of the model. The reconstruction error only tests how much information is lost in the mapping from high dimensional space to low dimensional space and back; it does not provide information about the semantics of the space, e.g. how similar are the executions of a particular grasp by different subjects.

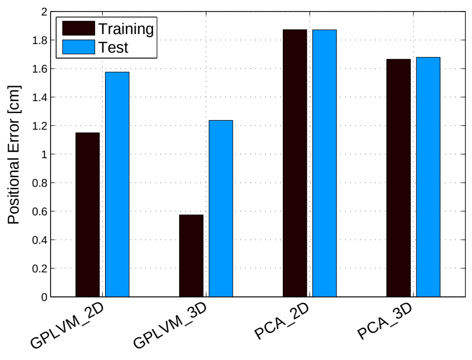
For all four data sets the positional (Figure 3.14a) and rotational (Figure 3.14b) errors were calculated.

Any model created with GP-LVM outperforms all models created by PCA in terms of reconstruction error. It is worth consideration the difference in performance between training and test data. In both GP-LVM models the training data has lower errors (both positional and rotational) compared to the test data. Interestingly such a trend is not visible for PCA where the error on training and test data are very similar. This seems to be due to the fact that the synergies from PCA tend to be over-smoothed, average trajectories (see Figure 3.15). Such average trajectories are “equally wrong” for training and testing. GP-LVM adapts better to the trajectories at the cost of a slight overfit. Nevertheless, the reconstruction error of GP-LVM is around 20% better than the error from PCA of the same dimensionality.

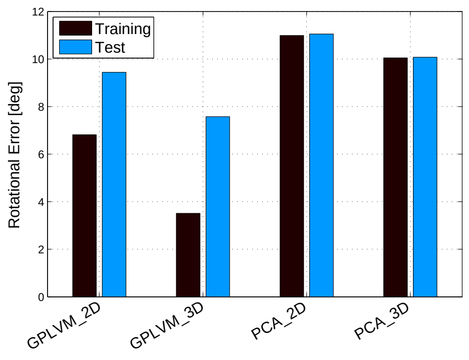
Increasing the dimensionality of the latent space allows to fit better the training data onto the manifold, decreasing the reconstruction error for the training set. Similar effects are observed for the testing set in this set, suggesting that the higher dimensional models are not overfitting the data. Importantly though, the decrease is significantly larger for the non-linear GP-LVM indicating that the correlations in the observed space are non-linear and cannot be modeled using PCA.

Both algorithms seem to treat positions and rotations with equal importance and in a similar fashion. If one compares the figures of positional (Figure 3.14a) and rotational (Figure 3.14b) the relative differences between models and test/training set are very similar. This is good news, as naturally 3D rotations are very difficult to visualize. So it seems that by visualizing only the positions is sufficient to get an idea on the model performance. One can therefore assume that the relative results will also be valid for the orientations.

Given a dimensionality, the overall performance of the extracted synergies is better for GP-LVM, as it has better results for the training set as well as for the test set. Human hand motion in general is nonlinear and therefore an algorithm that can cope with nonlinearities (such as GP-LVM) in the data will be superior. Additionally PCA looks for the largest variance in the data, which might be dominated by noise and the valuable information is blurred. GP-LVM, being a probabilistic approach, also has the ability to explicitly model the noise which increases the performance of the algorithm.



(a) Position error



(b) Rotation error

Figure 3.14: Reconstruction errors regarding the position and orientation of the training and test data sets.

3.6.2 Semantic Evaluation

Low reconstruction error is not the only desirable characteristic. The representation should capture and reflect semantic details of the data, like similarity among different users executing the same grasp or dissimilarity among different grasps.

3.6.2.1 Visualization of the mean grasp model

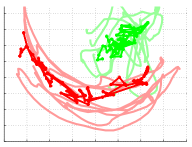
To access how well the GMM/GMR models fit to the original data, we project the latent trajectory of the GMM/GMR model back to the high dimensional space. The comparison to the original data gives insight on how good are the created GMM/GMR grasp models. We reduced the amount of data displayed for the sake of clarity. The movements are projected to the plane spanned by the palmar-distal directions. Figure 3.15 shows the fingertip movements of the index finger and the thumb projected onto that plane. The top image shows the corresponding grasp type as well as the plane the movements are projected onto. In the background of the other images the original movements of the 5 subjects is shown in a lighter color.

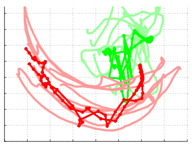
The grasp on the left side is a special variation of the power grasp, where the thumb is aligned with the axis of the cylinder. In this grasp type the thumb is relatively static, as it only abducts for the grasp. As abduction/adduction is a movement largely perpendicular to the plane, most of the movement is lost by to the projection. This makes the appearance of the thumb relatively random, but the 3D trajectory shows a distinct pattern.

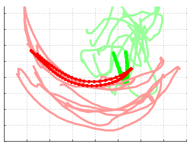
The grasp on the right side is a precision grasp, where the index and the thumb are used to pick up a small object. Therefore it is important that those two digits are close in the actual grasping phase. The background trajectories of the subjects clearly show that the (green) thumb and (red) index trajectories are very close.

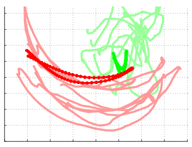
GP-LVM produces relatively rugged trajectories, but they follow the subjects trajectories quite well. They have roughly the same range of motion as the subjects. In the

(a) Grasp 4, Adducted Thumb

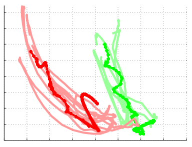

(b) GP-LVM 2D

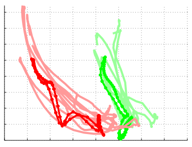

(c) GP-LVM 3D

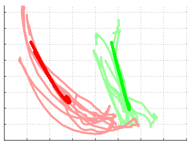

(d) PCA 2D


(e) PCA 3D

(f) Grasp 22, Tip Pinch


(g) GP-LVM 2D


(h) GP-LVM 3D


(i) PCA 2D

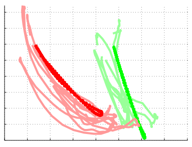

(j) PCA 3D

Figure 3.15: Projection of thumb and index fingertip position for Adducted Thumb (top row) and Tip Pinch (bottom row) grasps. Each row represents, from left to right, a picture of the grasp, and the projection of the mean from the GMM/GMR model computed with GP-LVM 2D, GP-LVM 3D, PCA 2D, PCA 3D. The red trajectories correspond to the index finger and the green trajectories correspond to the thumb. The lighter trajectories in the background are the original dataset.

precision grasp (right column) they reach a position where thumb and index finger are very close, which is a functional requirement of the grasp type. The three dimensional GP-LVM is smoother and it's trajectories fit the original ones even better.

PCA produces very smooth curves, but it cannot create the curved path of the subjects trajectories. There is an offset and the trajectory cannot follow the full motion amplitude of the subjects. When the dimension is increased to three, the shape of the trajectory improves – the curvature gets a little bit larger and the length of the trajectory better fits the subject's one.

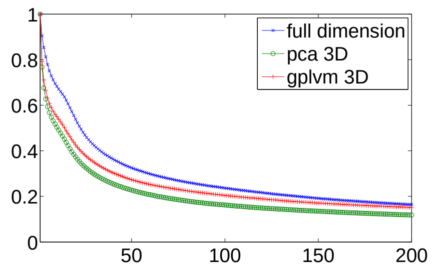
Overall GP-LVM outperforms PCA, since it is able to follow the path of the human fingertips much better for a given dimension. That comes at the cost of having more ragged trajectories. In most applications this is more desirable than having smooth trajectories, which follow the wrong path.

The rotational component of the fingertip cannot be easily visualized, so a comparable analysis on the rotations was not performed. Nevertheless it can be assumed that they will behave in a very similar fashion, as positions and rotations did in the reconstruction error (Section 3.6.1).

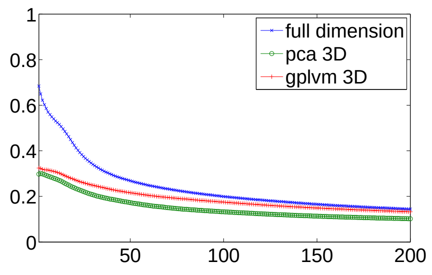
3.6.2.2 Local similarity of hand poses

Our first attempt to estimate the discriminative power of different synergic representations consists of inspecting which are the most similar poses for each time instance of a grasp.

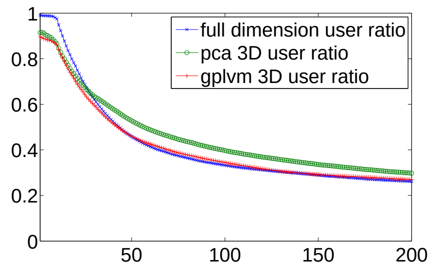
If each pose is represented by a point in pose space (either low or high dimensional), we can check which is the grasp class for the closest neighbors of each point, and compute the ratio of points belonging to the same grasp and the total number of points in the neighborhood. The results of such a comparison can be seen in Figure 3.16. This figure shows that, although the training data is more separable (in terms of grasp classes) in high dimensional space, it is preferable to classify new instances in low dimensional space. The nearest neighbor classification in the full dimensional data outperforms the ones in the low dimensional representation, the bottom row show us that it is due to two reasons. First, the fine details from the grasping phase are better preserved in full dimensional space. This can be seen as the big boost in performance for low dimensional representation when those frames are removed. The reason for this is that those details represent a small amount of variance of the data, and therefore they are removed in low dimensional representations. Second, the full dimensional representations of different trials from the same subject are quite similar, while different subjects do not contribute much to the classification. This can be seen in the graph 3.16d; the abrupt down slope for full dimensional data after the first 10 neighbors (all of them belonging to the same subject) means that GP-LVM3D has more correct neighbors between the 10th and the 20th than full dimensional data. This can be also observed in Figure 3.16c.



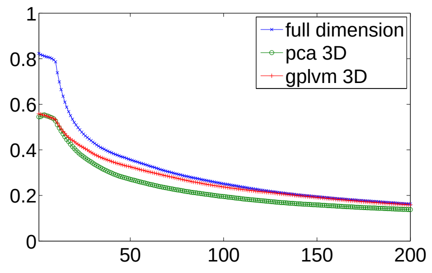
(a) Correct neighbors ratio for the training data.



(b) Correct neighbors ratio for the test data.



(c) Ratio of neighbors from same user at the grasp moment.



(d) Correct neighbors ratio for the test data at the grasp moment

Figure 3.16: The top row shows the correct neighbors ratio for increasingly bigger neighborhoods. For example, a value of 0.4 for a value in the horizontal axis of 50 means that 20 out of 50 neighbors were correct. The bottom row shows graphs corresponding to the 10 frames in the middle of each sequence (the grasp moment, see Figure 3.9).

3.6.2.3 Grasp similarity

The classification shown in Figure 3.16 does not take into account that a grasp is composed by a time sequence of points. Instead of classifying each individual time frame as belonging to one or another grasp class, it seems more natural to classify the whole sequence of poses that constitute a grasping action. Moreover, we would like to create a model for each grasp type, instead of representing it by the set of grasps that different subjects perform in the training phase.

In Figure 3.10 the evaluation of the trajectories from the dynamical model applied to the GP-LVM 2D representation is shown. Following the process depicted by Figure 3.13, each grasp model is based on five trajectories, as performed by the subjects. The dark line corresponds to the mean trajectory and the light area shows the variance the model has on certain points of the trajectory. One can clearly observe that different trajectories have a different signature in the latent space.

Computing the similarity between grasps (pose sequences) is not straightforward. We will use the probabilistic description of grasps in terms of Gaussian Mixture Models for this purpose. Based on these probabilistic models, we can compute how likely it is that each point x in the space is generated by a grasp g_i.

$$p(x|g_i) = \sum_{k=1}^{K} \pi_k^{g_i} \mathcal{N}(x|\mu_k^{g_i} \sigma_k^{g_i}) \tag{3.20}$$

$$p(g_j|g_i) = \prod_{\forall x \in g_i} p(x|g_i) \tag{3.21}$$

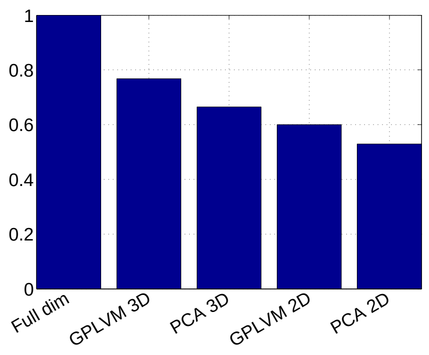
$$s(g_j, g_i) = (p(g_j|g_i) + p(g_i|g_j))/2 \tag{3.22}$$

Equation 3.20 states that the probability of a point x belonging to a grasp g_i is modeled with as a weighted mixture of gaussians, as explained in Section 3.5.5. We can compute the probability of a grasp g_j being generated by the model g_i as the product of the probabilities of each of its poses x being generated by g_i, once we assume these probabilities independent (Eq. 3.21). Other methods like HMM matching of sequences could be applied here instead, [52].

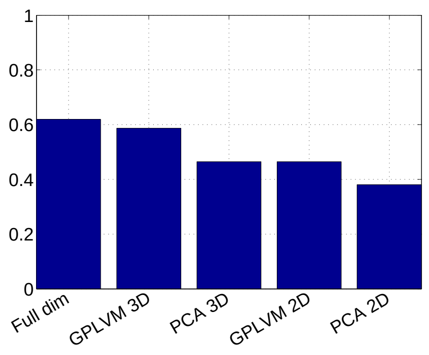
Note that this probability is not symmetric: $p(g_j|g_i) \neq p(g_i|g_j)$. We can define the similarity between two grasps $s(g_j, g_i)$ as the average of those two quantities, Eq. 3.22.

Following these equations we can compute the probability of a new grasp sequence having been generated by a particular GMR model. By comparing those probabilities we can estimate which is the most likely grasp class that generated that sequence, and compare it with the real grasp that was actually executed. Figures 3.17 show this classification rates for models composed by $K = 3$ and $K = 6$ Gaussians, in which the training data or the test data is classified.

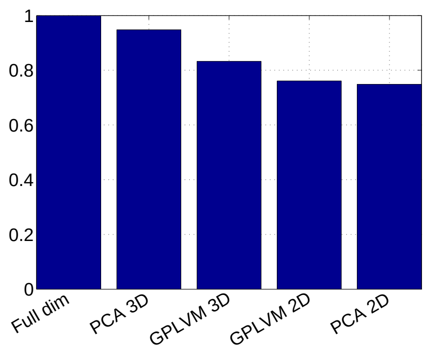
Let us first concentrate on how different dimensionality reduction methods behave for a given number of Gaussians in the Gaussian Mixture Model. In Figure 3.17a the high dimensional space dominates, meaning that the data is perfectly separable in high dimensional space. By employing postural synergies we are losing the ability to perfectly sep-



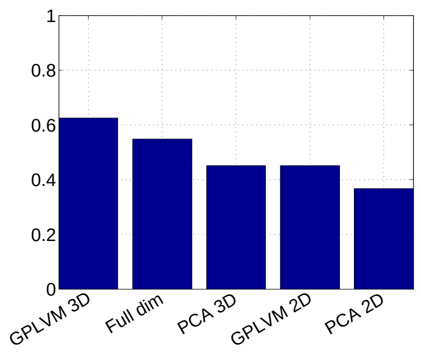
(a) training data 3 Gaussians



(b) test data 3 Gaussians



(c) training data 6 Gaussians



(d) test data 6 Gaussians

Figure 3.17: Classification rate for GMM/GMR models with 3 (top row) and 6 (bottom row) Gaussians. In the left column the data used for creating the models was used also for testing it, while in the right one a new set of data was used. GP-LVM outperforms PCA for a given dimensionality, and performs similarly to the full dimensional model, which uses between 5 and 10 times more dimensions. Although the grasp training set is more separable in PCA3D and full dimensionality than in GP-LVM3D when 6 Gaussians are used, the latter generalizes better over new data and therefore outperforms the rest when classifying previously unseen data.

arate training data. Lower dimensional spaces (GP-LVM 2D and PCA 2D) have more trouble separating the data than higher dimensional ones (GP-LVM 3D and PCA 3D). However, the classification changes substantially when the data to be classified has not been taking into account for creating the models. The high dimensional space is not able to perfectly model the new data since it falls much further from the mean of the model than any previously seen sequence. We can conclude that the full dimensional model is not able to model properly the variance of the data; it assumes that it is much lower than it actually is. The drop in performance for the lower dimensional spaces is much smaller than for the full dimensional case. The performance of GP-LVM 3D is comparable to the one of the full dimensional case, while using a more than 10 times smaller representation. The 3D spaces are still outperforming the 2D, showing that an additional synergy helps to separate grasps that could not be differentiated with a 2D model.

With respect to the importance of number of Gaussians in the mixture model, we can observe in the low part of Figure 3.17 that, despite having an impact in the performance of training data classification, it is mostly irrelevant when it comes to generalizing over new sequences. The reason for this is that the model overfits the training data, without improving significantly the performance over unseen sequences.

3.6.3 Data-driven Taxonomy

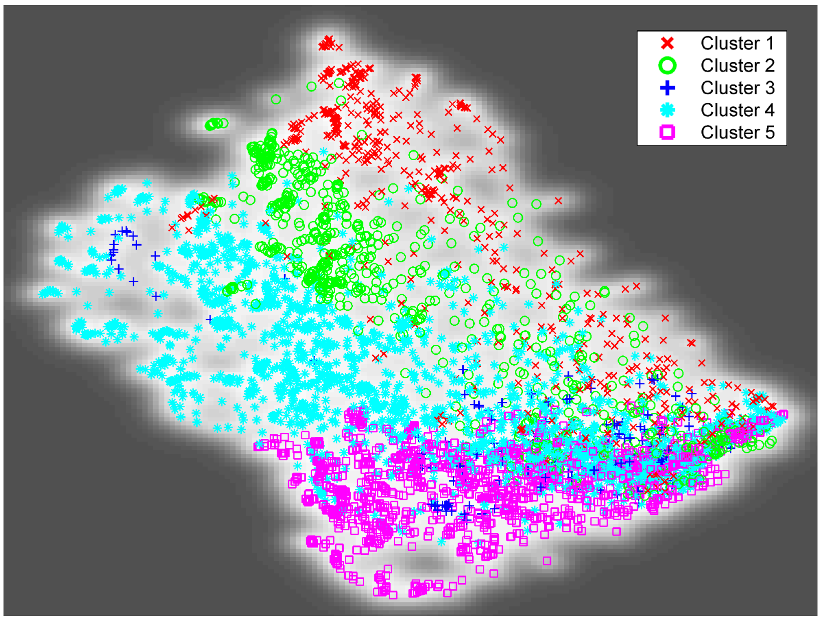
We performed average linkage clustering (from the Matlab Statistical Toolbox, also known as UPGMA) based on this similarity measure. The result of the algorithm can be seen in Figure 3.18a.

The number of clusters was chosen to be 5 since further subdividing the clusters overfits the data, i.e. *cluster four* was split into two groups with similar characteristics. Reducing the number of clusters resulted in large, non-descriptive clusters.

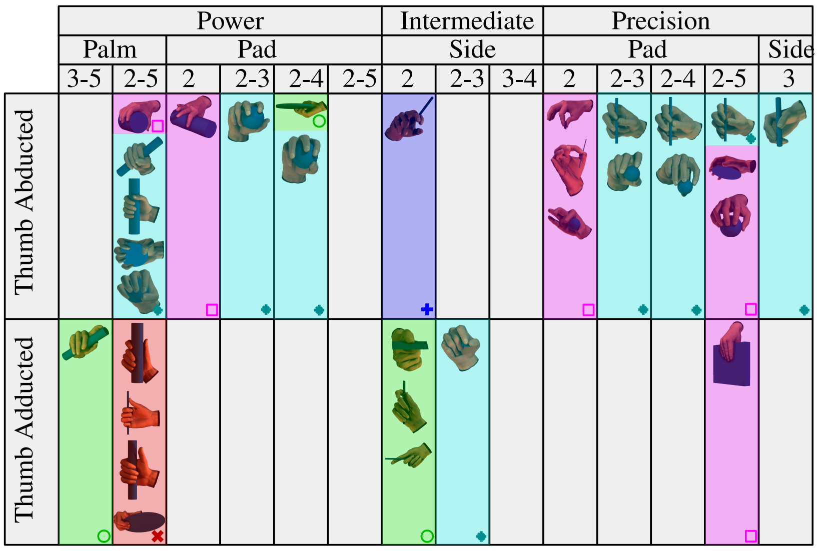
The relation between the clusters extracted from data and the taxonomy described in [11] is shown in Figure 3.18b. The comparison show us some similarities and dissimilarities between the theory-based and data-based taxonomies. First, both of them agree in the principal role of the thumb in the grouping of clusters. The taxonomy in [11] theorized the existence of two big groups of grasps based on the position of the thumb: abducted and adducted thumb. The data-driven taxonomy complies substantially to such division: abducted thumb complies 92% of Cluster 4 and 87% of Cluster 5 (and the single grasp in Cluster 3); adducted thumb is composed by the totality of Cluster 1 and 80% of Cluster 2. It is remarkable also that all the grasps in Cluster 1 are Power grasps with palm contact where all fingers but the thumb act in unison (there are only two virtual fingers, see [11] for more information).

One remarkable dissimilarity is the low correlation between the clusters and the classification in terms of palm contact with the object: Power, Intermediate and Precision. This might be a consequence on the pose data: it is very difficult to asses contact between palm and the object given only the fingertip position and orientation of the fingertips with respect to the wrist. For example, the differences between *sphere 4 finger* (power grasp), *lateral tripod* (intermediate grasp) and *quadpod* (precision grasp) are really subtle in terms of the sensing we acquired.

Nevertheless, this alignment of the data-based and theory-based taxonomy is promising. The inclusion of data regarding contacts between hand and object would potentially improve such alignment further.

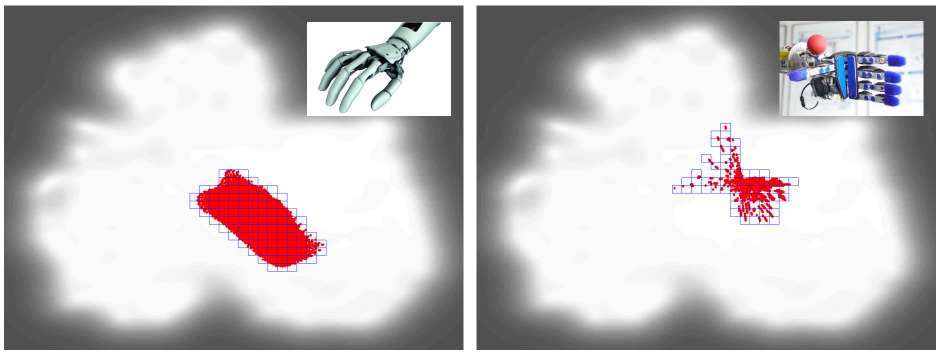


(a) Hand poses clustering in low dimensional space


Power
Intermediate
Precision
Palm
Pad
Side
Pad
Side
3-5
2-5
2
2-3
2-4
2-5
2
2-3
3-4
2
2-3
2-4
2-5
3
Thumb Abducted
Thumb Adducted

(b) Taxonomy colored according to clusters

Figure 3.18: The two-dimensional models of grasps (Figure 3.10) extracted with GP-LVM were used to group grasps according to their similarity (Eq. 3.20-3.22), as can be seen in the top figure. The bottom figure shows how those clusters (shown as color overlays on the table) align with the manually designed taxonomy described in [11].



(a) 5 DOF Michelangelo (b) 8 DOF ARMAR hand

Figure 3.19: Movements of Michelangelo Hand [155] and ARMAR hand [25], mapped onto the low-dimensional space spanned by human hand grasping. The Michelangelo hand was enhanced with three extras degrees-of-freedom from the initial design with two, evaluating in this way the suitability of those changes. Although the ARMAR hand has three degrees more than the Michelangelo, the latter covers a larger area. This mean that it can reach a larger portion of the human grasping poses and therefore it is more anthropomorphic for grasping purposes.

3.6.4 Hand prosthesis design using grasping synergies

In the context of manipulators hardware, we would like to build artificial hands that share their dexterity with that of a human. Due to engineering limitations, realizing such design is currently infeasible. However, by gaining knowledge of the intrinsics in the grasping process we can assist and motivate hardware decisions based on grasping principles rather than in the ad-hoc manner which is common practice. More precisely, the insight about low-dimensionality control of human grasps seems particularly attractive for manipulators design, given that manipulator complexity is not always beneficial [137, 166]. Work on manipulator underactuation [135] and manipulator design optimization [55, 57] shows a trend on improving grasping capabilities in the design phase of the manipulator, which can safe computational power in the grasp execution phase.

The methodology explained throughout this chapter allows us to define, in a data-driven way, what are the “ingredients” which constitute human dexterity. The manifold created from human grasping data represents in a compact way all the poses needed by a human to perform any kind of grasp (if we consider that the taxonomy in [11] is complete). Such manifold can be used therefore to inspect the anthropomorphism of robotic hand designs; the more anthropomorphic a manipulator is, the more fingertip positions will be able to imitate and therefore the bigger portion of the manifold will cover by moving its actuators.

In our collaborative work [14], T. Feix uses this concept to evaluate different robotics and prosthesis hands for the purpose of designing more anthropomorphic hands. The system allows the user to expand current designs with new degrees of freedom, vary the lengths of the links, or change completely the manipulator configuration and evaluate instantaneously the anthropomorphism of the hand. The evaluation is done terms of percent of the space spanned by the human covered by the artificial hand (Figure 3.19).

3.7 Summary

Postural synergies as a modeling paradigm has been commonly used in the robotics community over the last decade. The technique has been exploited for a large range of different applications to improve control and increase our understanding of the complex relationship between the joints and muscles in our hands. However, the methodology used for extracting these synergies have been limited to linear spectral methods (namely PCA), without explicitly stating the consequences that this choice had in terms of mapping restrictions. This makes itself evident as the original approach to extract the representation using principal component analysis have within the robotics community become synonyms with the method itself and not as one possible route to reach a synergic representation. We argue that this significantly limits the usefulness of postural synergies as a modeling paradigm.

In this chapter we have tried to provide an understanding of the implications and limitations of the original approach to extract postural synergies. We show that by leveraging recent development from the machine learning community and by careful consideration of the data significant improvements of the approach can be achieved. We have presented both qualitative and quantitative results for applying two different approaches for learning low-dimensional representations of hand pose data. Doing so we show that application of modern non-linear dimensionality reduction results in an improved representation, indicating that the joint correlations in a hand is non-linear. Further, we have provided intuition into extracting and modeling grasp data using low-dimensional representations. The intuitions we provide are general and applicable to any problem setting with dynamic data.

3.7.1 Future work

The work presented in this chapter helps us understanding how humans execute different grasps, how do their hand poses evolve in time, how similar are different grasps, etc. As mentioned previously, this has the potential to improve a grasp-by-demonstration system both from the human analysis side (improving the estimation of the human hand pose) as in the robot side (defining control laws that resemble more human grasps, building more anthropomorphic hands, etc). However, the integration of the synergies with the rest of the modules in the grasping system has not been performed under the scope of this thesis. The reasons for this are two-fold.

First, the grasp models and the hand pose estimation system use two different high dimensional spaces: hand pose estimation is based on joint-space data, while the synergies are defined on task-space data. There are two possible ways of solving this mismatch: to make the hand pose estimation work in task-space data, or to translate the results of the grasp models into joint space data. The first solution is promising, since task-space representation of the human hand pose seems more suitable, although higher dimensional. The synthetic model of the hand was adapted to provide task-space information, and therefore this line of work will be exploited in the near future. The second possibility, translating task-space data into joint-space, involves the classical problem of inverse kinematics. Performing inverse kinematics from real human hand data is far from trivial, given that the kinematic models of the human hand are usually inaccurate and more rigid than the actual hand.

The integration of the models with the robotic execution is related to the correspondence problem, treated in Chapter 4. The analysis performed in Section 4.2.1, based on the concept of Virtual Fingers, could be adapted to the models obtained here once the absence of data related to the contact between hand and object is solved. This remains as future work.

Finally, another line of work that can be continued from this chapter is related to shaping the grasp manifolds depending on the future usage of the models. One of the GP-LVM strenghts is the possibility of integrating prior knowledge about the training data that shapes accordingly the low dimensional space. For example, if the goal of the low dimensional space is to discriminate between grasps, the optimization procedure described in Section 3.2.2 can include terms that penalize representations in which training data from different grasps are far from each other. Such priors could encode also information about temporal continuity, for example. Experimenting with such priors would provide us better models for specific applications.

Chapter 4

Human to robot mapping

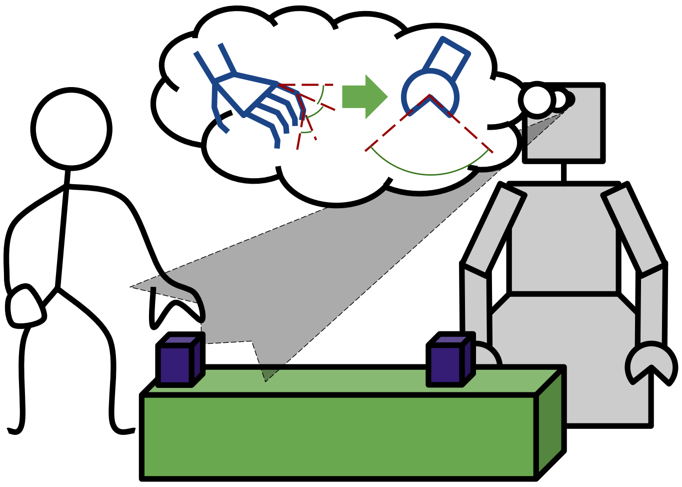


Figure 4.1: After having extracted the hand pose of the human teacher, the robot has to compute an equivalent pose for his own embodiment. The number of parameters and/or their effect on the pose can differ substantially between robots and humans.

The first requirement in a learning by imitation environment is a sufficient degree of similarity between the learner and the teacher, [147]. Without this, imitation might become impossible; a car cannot imitate a persons facial expression. However, the vagueness of the requirement should be apparent to the reader; first, the notion of correspondence is highly subjective, and second, "sufficient" is not an absolute measure of correspondence. Actually, this subjectivity makes very difficult to pronounce categorical statements as the one in the beginning of the section (see Figure 4.2).

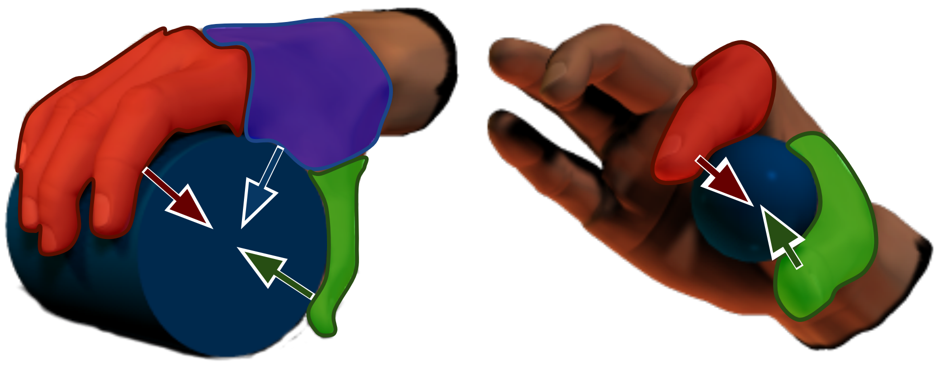
As it has been mentioned in Chapter1, *Demonstrations* have a trivial embodiment map-

Figure 4.2: Mapping plausibility is highly subjective [160]

ping of state-action $g_E(z,a) = I(z,a)$. This happens when the embodiment of the leaner is identical to the one from the teacher, plus the learner executes its actions in an environment identical to the teacher's environment. However, this condition is very restrictive. In order to deal with differences in the embodiments, the embodiment mapping $g_E(z,a)$ should try to maximize a correspondence measure which compares the *effects* of the actions on the environment and the body of the self, [147]. Formally, we define the effects as X, which can integrate the states Z, actions A, and maybe higher level concepts as goals. Then, the correspondence measure can be defined as $d : X \times X \rightarrow R$.

The learners with the kinematic structure most similar to humans are usually found in computer simulations. These models are used mostly in research focused in task-learning aspects, such as learning controllers for human grasping [162] or learning representation of full body actions [210]. The mapping in these cases is close to trivial since it involves either disregarding degrees of freedom from the teacher (like the hand joints in full-body action learning) or setting to zero some joints in the learner model (because virtual models usually treat all joints as ball joints with 3 degrees of freedom, while many joints in humans have one or two degrees of freedom only). The most common correspondence measure in these cases in root mean square (RMS) error of the joint angles that were actually mapped, or RMS of end-effector positions (such as feet and hands in full body problems).

The design of humanoid robots mimics the kinematic structure of humans. However, they have physical limitations inexistent in computer simulations. Therefore, it is natural that their embodiment mappings can be more complex than the ones described above. A common assumption in humanoid robotics is that *humanoid robot kinematics can be embedded into human body kinematics* [208], i.e. humanoid kinematics can be considered a subset of human kinematics. In this case, the learner can reproduce the teacher movements either by direct mapping of joint angles [111] or by optimizing the joint angles to make some keypoints in its body match similar point in the teacher's body after some scaling [208]. The similarity between humanoids and humans in terms of hand kinematics is very low though. The complexity of human hand kinematic structure [179] and specially the thumb [92] forces the humanoid designers to use very simplified versions of the human hand. Some of these robots treat the wrist as a rigid object and simply attach objects if manipulation is required [111, 49]. When humanoids do have actuated hands, they usually have a lower number of fingers [19, 103, 95, 90]. An exception is ARMAR-IIIb [25], which has a hand model with five fingers and 8 degrees-of-freedom. However, even when hands resemble the outer appearance of human hands, their movement capabilities differ considerably from the human ones. For example, ARMAR fingers are driven by fluidic actuators which make the finger movement very fast, but only allows them to be completely flexed or straight [180]. Due to all these factors, the embodiment mapping for real robots cannot be treated as trivial.



(a) Three Virtual Fingers for Large Diameter Grasp (b) Two Virtual Fingers for Inferior Pincer Grasp

Figure 4.3: Virtual fingers are groups of fingers (and/or palm) which act in unison, i.e. whose contact points are close to each other and whose force orientation is similar. In the picture we see virtual finger groups overlaid on the real fingers, together with the force exerted by each virtual finger.

A system which addressed explicitly the problem of embodiment mapping for grasping can be found in Kang et al., [119, 120]. This work defines a correspondence measure which penalizes imitations which diverge from the teacher in terms of applied forces. However, the measure is robust to different embodiments by using the concept of *Virtual Fingers* [20]. Fingers are grouped into *Virtual Fingers* which act in unison, creating in this way "functional units" which can actually be implemented by a different number of real fingers (Figure 4.3). For example, while five fingers are used when grasping a cylinder like in Figure 4.3a, there are only three opposing forces, and therefore only three virtual fingers (one composed by the palm, another by the thumb and the third one composed by the rest of the fingers, which exert parallel forces on the object).

The system described in this chapter uses a correspondence measure motivated by the concept of virtual fingers as well. However, such measure is not applicable in our *External Observer* setup, where we have no access to the forces the teacher exerts on the object. Therefore, we simplify the mapping to a classification-based approach similar in spirit to the one described in [78], but then evaluate such mapping in simulation with the virtual fingers correspondence measure. We classify the human grasp in terms of grasp class, [11], and approach (side-grasp or top-grasp). Apart from the simulation evaluation, the method was tested in two real platforms: a grasping setup with a KUKA arm and a Barrett hand and ARMAR humanoid platform.

The remaining of the chapter is organized as follows. In Section 4.1 we describe the details of the grasp classification and mapping. Then we evaluate our method in Section 4.2, both quantitatively in simulation 4.2.1 and qualitatively in two different robotic systems: an industrial setup with a KUKA arm and Barrett hand (Sec. 4.2.2), and the humanoid robot ARMAR (Sec. 4.2.3). Finally, we summarize the work in Section 4.3, and discuss future directions in Section 4.4.

4.1 Methodology

The system described in this chapter is divided in three parts: grasp classification, grasp correspondence and grasp planning. Grasp classification is largely an adaptation of our hand pose estimation system, described in Section 2.4. Instead of providing the pose of the human hand, it outputs grasp classes and discrete grasp orientation (side or top). Grasp correspondence is a discrete set of correspondences between human and robot grasp classes, manually defined. Finally, grasp planning describes how different robotic grasp classes trigger different kinds of grasp executions related to both their approach and grasping phase.

4.1.1 Grasp classification

In Section 2.4 we have described how the hand pose, in terms of joint angles, can be extracted from a sequence of monocular images of a human grasping task. The most straightforward way of using this information would be to try to match the fingertip positions of the robot to the fingertip positions of the human through inverse kinematics. However, this approach has some disadvantages. First, since the embodiments can be very different, such a mapping can be inefficient even for perfect positioning of the fingertips (e.g. the success of the grasp relies on some characteristic inexistent in the robot, like high fingertip friction). Second, the fingertip positioning is bound to be only an approximation of the human one, and the effect of the position error might cause grasp failure (e.g. incorrect positioning of a fingertip in a pinch grasp will likely cause torques in the object). Kang et al. addresses this problem by using of *Virtual Fingers*: real fingers are arranged into groups that act in unison, i.e. they have contact points close to each other and similar force directions (Figure 4.3). However, this approach needs knowledge about the contact points between the demonstrator's hand and the object, which is difficult to obtain out of simulated environments.

We opted for using an example-based mapping of grasps. Instead of trying to mimic the posture of the demonstrator at every moment, we treat the whole grasping sequence (approach and grasp) as a task unit which is recognized first and then executed in a way that fits the particular hand executing the grasp.

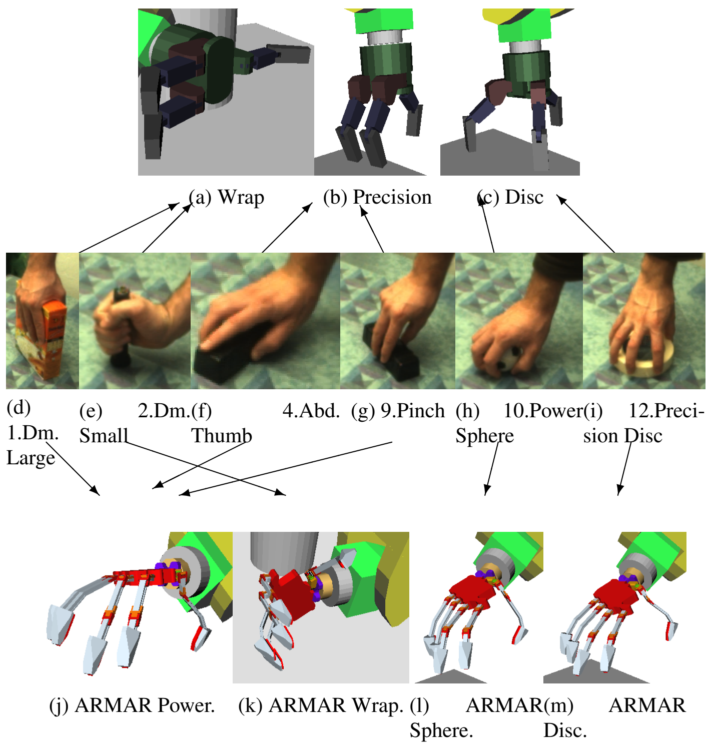


Figure 4.4: The six grasps according to Cutkosky's grasp taxonomy [62] considered in the classification, and the three grasps for a Barrett hand and ARMAR hand, with human-robot class mappings ((d,e)→(a),(f,g)→(b), (h,i)→(c)), (d,f,g)→(j), (e)→(k), (h)→(l), (i)→(m) shown. a) Barrett Wrap. b) Barrett Precision. c) Barrett Precision Disc d) Large Diameter grasp, 1. e) Small Diameter grasp, 2. f) Abducted Thumb grasp, 4. g) Pinch grasp, 9. h) Power Sphere grasp, 10. i) Precision Disc grasp, 12. j) ARMAR Power, k) ARMAR Wrap, l) ARMAR Sphere, m) ARMAR Disc

Our mapping requires the grasp class $\hat{y}$ as well as the hand orientation with respect to the camera $\hat{o}$. We used the system described in [8], which is a predecessor of the one described in Section 2.4. This system uses a smaller database which contains only the six grasp types from the Cutkosky taxonomy [62] (see Figure 4.4). Apart from this, the main difference is that it does not exploit any temporal consistency model (see Section 2.4.4.2), and uses Euler angles instead of Rotation matrices for the representation of $\hat{o}$. The class of the grasp in each time-step is extracted easily from the database entry correspondent to the most likely pose.

The perception system was updated and adapted for the integration with ARMAR III-b, as it is described in Section 4.2.3).

4.1.2 Grasp correspondence

The estimated grasp class as well as the orientation and position of the hand and the object are used to instantiate a robot grasp strategy. The human-to-robot grasp mapping scheme is defined depending on the type of robot hand used. The Barrett hand is a three fingered, 4DOF robotic hand with an embodiment substantially different to a human hand. ARMAR hand is a five fingered, 8DOF robotic hand with an embodiment similar to a human hand. The preshapes used for the hands are shown in Fig. 4.4. There are three preshapes for the Barrett hand:

- Barrett Wrap: used for grasps with a preshape with large aperture, like Large and Small Diameter grasps;
- Barrett Precision Grasp: for small aperture preshapes like the Pinch grasp and the Abducted Thumb (executed as a pinch grasp due to the hand kinematic constraints);
- Barrett Precision Disc: for circular objects.

There are four preshapes for the ARMAR hand:

- ARMAR Power preshape is applied for grasps with four parallel fingers and thumb opposed to them, like Large Diameter, Pinch and Abducted Thumb.
- ARMAR Wrap is applied for the Small Diameter where the thumb is not opposed to the rest of the fingers.
- There are two preshapes for round objects, ARMAR Sphere (for Power Sphere) and ARMAR Disc (for Precision Disc); the differences are in the pose of the thumb (more opposed in the Disc) and in how straight are the rest of the fingers (more bent in Power Sphere).

The grasp mapping is performed as shown in Algorithm 1. The hand orientation estimate $o_{h\to c}$, along with the hand position estimate $p_{h\to c}$ and the estimated position and orientation $o_{o\to c}, p_{o\to c}$ of the grasped object all relative to the camera are used to derive the estimated position and orientation of the human hand relative to the object $o_{h\to o}, p_{h\to o}$. The hand orientation relative to the table plane o_h is extracted from $o_{h\to c}$ and the orientation of the camera o_c, obtained through the robotic head kinematics. The object position and orientation is assumed known.

4.1.3 Grasp planning

According to [114], a grasp action involves two main functions: the approach component (involving the arm muscles) and the grasp component (involving the hand muscles). Although it has been showed that these systems are closely correlated, people focused mainly on one of the two subsystems. There are systems performing imitation of the arm [37] or, more generally, the upper-body [30], as it will be shown in Section 4.2.3. The arm/upper-body imitation does not experience the self occlusion to the same extent as the hand does.

The system first decides which preshape to use based on the recognized grasp. Then, the approach vector is chosen. Two different ways of approaching the object are used, based on the orientation of the human hand; if the palm orientation o_h is similar to the

```
Data: Human Grasp $G_h$, $p_{h\to o}$,$o_{h\to o}$, $o_h$
/* Robot Hand $H \in \{Barrett, ARMAR\}$                                   */
/* Robotic Grasp $G_r$                                                     */
/* Approach Vector $a$                                                     */
/* Distance palm-fingertip $\delta$                                        */
if $H = Barrett$ then
    if $G_h \in \{LargeDiameter, SmallDiameter\}$ then
        $G_r = BarrettWrap$;
    else if $G_h \in \{Pinch, Abducted\}$ then
        $G_r = BarrettPrecision$;
    else if $G_h \in \{PowerSpherical, PrecisionDisc\}$ then
        $G_r = BarrettPrecisionDisc$;
else if $H = ARMAR$ then
    if $G_h \in \{LargeDiameter, Pinch, Abducted\}$ then
        $G_r = ARMARPower$;
    else if $G_h = SmallDiameter$ then
        $G_r = ARMARWrap$;
    else if $G_h = PowerSphere$ then
        $G_r = ARMARPowerSphere$;
    else if $G_h = PrecisionDisc$ then
        $G_r = ARMARPrecisionDisc$;
if $o_h = o_{side}$ then ;                              /* see Fig. 4.4e */
    $a = a_s$;                                     /* approach from side */
else
    $a = a_v$;                                      /* approach from top */
Set hand to preshape $G_r$;
Set hand to orientation $o_{h\to o}$;
Approach following $a$ towards $p_{h\to o} - \delta$;
Grasp;
```

Algorithm 1: Pseudo-code for grasp mapping.

one in Fig. 4.4e the object is approached from the side, otherwise it is approached from the top. Based on the estimated type of grasp, the system differentiates between volar and non-volar grasp, [119], i.e., whether there should be a contact between the palm and object or not. The original volar grasps are the Large Diameter, Small Diameter, Abducted Thumb and Power Sphere grasps, see Fig. 4.4. However, the limitations of the hands embodiments make the usage of the palm in the Abducted Thumb and Power Sphere grasps impossible. In a human Abducted Thumb grasp the palm adapts its shape to the object, and the abduction/adduction degrees of freedom of the fingers are used; the robotic hands studied here lack those degrees of freedom, so the Abducted Thumb is mapped to a Pinch Grasp. In the case of the Power Sphere, the robotic hands cannot apply a volar grasp due to the larger length of robotic fingers.

The volar grasping is performed in the following order:

1. The robot adopts the hand orientation and preshape corresponding to the estimated human grasp.
2. The robot hand approaches the object centroid until it detects contact on the palm sensor. After that, it closes the hand.

4.2 Evaluation

Evaluation of imitation systems is challenging, since the measure of similarity is, as it has been argued in the introduction to this Chapter, inherently subjective. In order to evaluate our discrete mapping, we defined a correspondence measure based on where groups of functionally similar fingers (virtual fingers) were positioned on the object. The results of this metric are shown in simulation since contact points on objects could not be measured in real experiments. These experiments are continued in Section 5.1, where online tactile feedback is used to correct errors in perception. Real experiments were run in two scenarios. First, a scenario equivalent to the simulation one (with the Barrett manipulator) was used as a "proof-of-concept" to show that the approach can be implemented in a real setup. Second, the grasp mapping system was ported to ARMAR-IIIb and other software modules for full robot mapping in collaboration with Martin Do et al. from Karlsruhe Institute of Technology, KIT.

4.2.1 Correspondence metric and virtual grasping

Evaluating the performance of a grasp imitation system is not trivial. Grasp stability is not the only goal of an imitation system, since in such a system we aim towards performing stable grasps as similar as possible to the demonstration performed by the teacher. Direct comparison of joint angles or end-effector positions is not possible given the differences in embodiments between teacher and learner.

We have decided to compare the grasps using a correspondence metric based on the concept of virtual fingers ([20]), computed based on the equations stated in [118]. Virtual Fingers offer us a functional representation of the grasp which can be computed for any hand. Therefore, the virtual fingers provides as a common space where we can compare grasp executions from different hands such as the human, ARMAR and Barrett hand.

As cited at the beginning of the chapter, a virtual finger is a group of real fingers (including the palm) that act in unison. Therefore, functionally-similar grasps should have similar position and orientation of the virtual finger contacts.

The virtual finger computation tries to minimize two factors: the number of virtual fingers N, in order to achieve a compact representation, and the heterogeneity of the real fingers R_i conforming each virtual finger V_k, described as a cohesive index for virtual finger k, C_{V_k}. The cohesive index of each virtual finger is computed based on the degree of force coupling (cosine of the angle between the forces) between each two forces f_i, f_j applied with any of the fingers within a virtual finger:

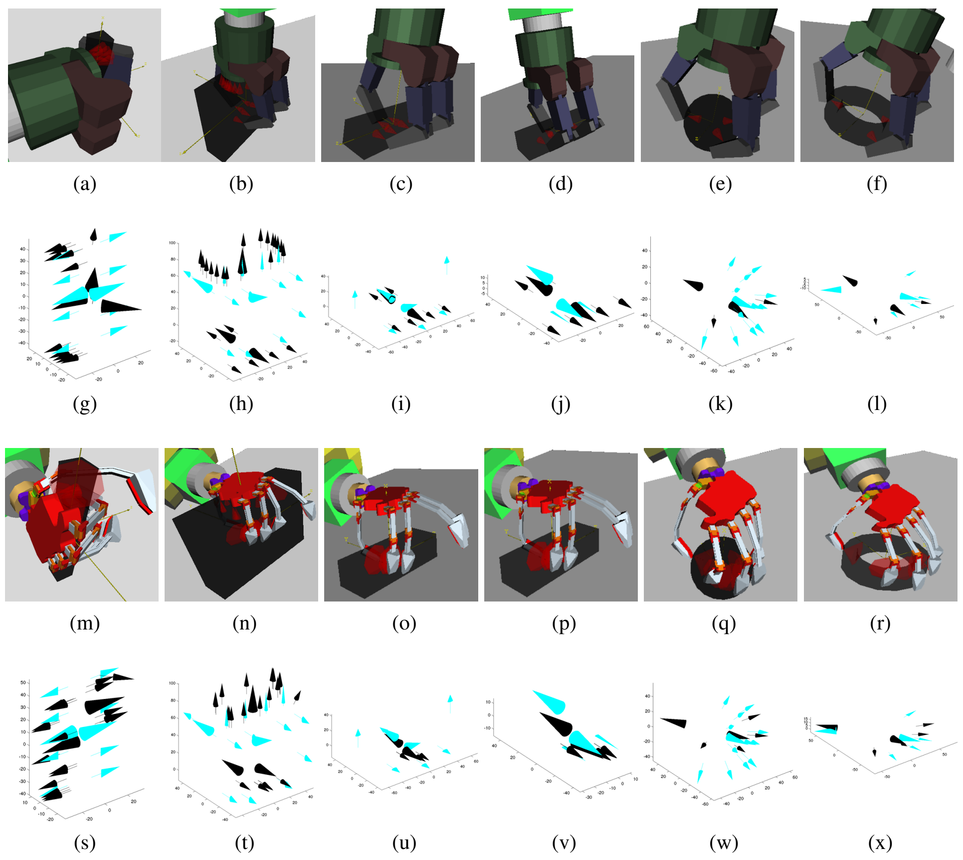


Figure 4.5: First and third row shows the grasp execution in absence of errors for Barrett and ARMAR Hand. Second and forth row show a comparison between the contacts for Barrett(black)-Human(blue) hands and ARMAR(black)-Human(blue) hands. The big arrows show the average pose of the virtual fingers,$\mathcal{P}_{V_k}$

$$
\begin{aligned}
D_c(i,j) &= \frac{f_i \cdot f_j}{|f_i| \cdot |f_j|}, \quad m_{ij} = \frac{1 + D_c(i,j)}{2} \\
C_{V_k} &= \prod_{\substack{i \in R_i, j \in R_j \\ R_{i,j} \in V_k}} m_{ij}^{\xi}, \quad \xi = \binom{F(V_k)}{2}^{-1}
\end{aligned}
$$

where $F(V_k)$ is the number of fingers within V_k. For example, if all forces within a virtual finger k are parallel, $C_{V_k} = 1$. If any two forces belonging to the virtual finger are perpendicular, $C_{V_k} = 0$. So, in order to find the best configuration of virtual fingers, we maximize the cohesive indexes C_{V_k} trying to keep the number of virtual fingers small:

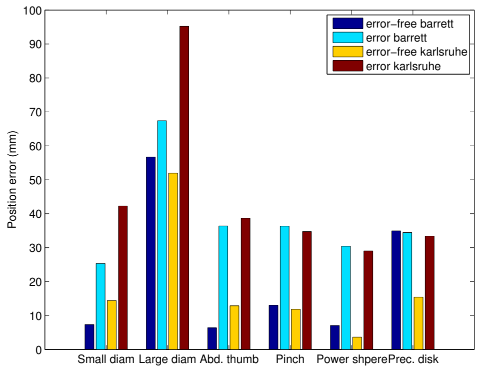
$$\begin{array}{ll} \text{Maximize} & C_{eff} = (\frac{1}{N!}\prod_{i=1}^{N} C_{V,i})^{\frac{1}{N}} \\ \text{Subject to} & N \in 1,2,3,4,5,6, \ \bigcup_{i=1}^{N} V_i = R \\ & V_i \cap V_j = \emptyset, i \neq j, 1 \leq i,j \leq N \end{array}$$

For every possible combination of real fingers R_i assignment to virtual fingers V_k, the coefficient C_{eff} is computed. The assignment with the highest C_{eff} is selected as the virtual finger representation of the grasp. The position and orientation of contacts is automatically extracted from the robotic simulator for the robot grasps, and it was tagged manually from images for the human grasp.

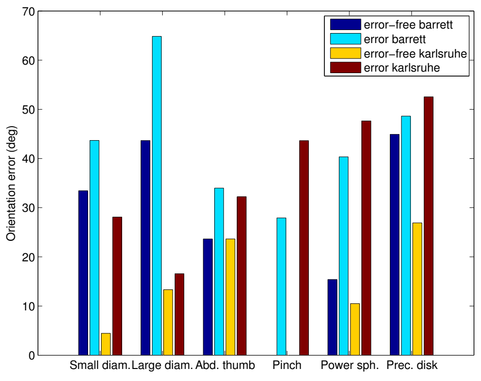
So far, all the real fingers R_i have been assigned to a virtual finger V_k. In order to compare the configuration of different hands we will define the contact pose (6d, including position and orientation) $\mathcal{P}_{V_k}$ of a virtual V_k as the average of the contact poses $\mathcal{P}_i$ within this virtual finger:

$$\begin{aligned} \mathcal{P}_{V_k} &= \frac{1}{T(V_k)} \sum_{\substack{i \in R_i \\ R_i \in V_k}} \mathcal{P}_i \\ T(V_k) &\equiv \text{number of contacts within } V_k \end{aligned}$$

In the first experiment, perfect object pose estimation and perfect hand pose recognition is assumed. Fig. 4.5 represents the grasps and the contact comparison between the robotic hands (black) and the human hand (blue). The big arrows show $\mathcal{P}_{V_k}$, and the small arrows show all $\mathcal{P}_i$. It can be seen in this figure that the pose of the virtual fingers and even the number of them does not coincide always. For example, the Barrett hand has three virtual fingers for the Small Diameter grasp, while Human and ARMAR hands have two (Fig. 4.5a,g,m,s). The reason for this mismatch is that the fingers of the Barrett hand are longer than the ARMAR fingers, and longer than the fingers from the human demonstrating the grasp, so the object is touched by the last phalanx in the edges instead of the face. Another significant difference in the number of virtual fingers appears in the Power Sphere grasp (Fig. 4.5c,k,q,w). The human grasp has just one virtual finger, while the robotic hands have two. For the human, placing the thumb opposite to the rest of the fingers is uncomfortable. The big contact surface and therefore big friction between the hand and the ball allows him to place the fingers in a relatively unstable way. However, the contact surface between the robotic hands and the object is much smaller, so the thumb should be placed in opposition to the rest of the fingers. In contrast, for the Precision Disc grasp (Fig. 4.5f,r,l,x) the human needs to place the thumb in opposition to the rest of the fingertips due to the lower friction between the hand and the object. It is also interesting to reason about the results for the Large Diameter grasp (Fig. 4.5b,h,n,t). Apparently there is a big difference between the average virtual finger position, but the actual contacts look similar. The reason is that the human fingers have contacts in both the proximal and distal phalanges, while the robot achieves a contact just in the distal "phalanges". Finally, it should be pointed that despite that it is impossible to imitate the Abducted Thumb grasp properly, the position of the virtual fingers is quite accurate, with a deviation in orientation in one of them due to the two contacts from the human palm and index fingertip in the top of the object, inexistent in the robotic grasps.



(a) Position error



(b) Orientation error

Figure 4.6: Error in position (mm) and orientation (degrees) for each of the six grasps tested (Small Diameter, Large Diameter, Abducted Thumb, Pinch, Power Sphere, Precision Disc). The different bars represent the error done by barrett hand without and with initial error of 5cm, and ARMAR hand without and with initial error of 5cm.

We present a more concise view of the experiments in Figure 4.6: it represents the average error in virtual fingers position and orientation (position and orientation component of $\mathcal{P}_{V_k}$) between the robotic hands (where Barrett error is represented in black and ARMAR in white) and the human hand. However, it should be noted that this measure is a lower bound of the error: in cases where the number of virtual fingers is different, this represents the average distance between the best matching virtual fingers. Sometimes this mismatch is known and natural (like the different number of virtual fingers in the Power Sphere grasp), but sometimes this means that one finger failed to touch the object. This happens mainly in the experiments with position error with the ARMAR hand, and will be addressed later. For the case of perfect data (blue and yellow bars in Fig. 4.6) we can draw a number of conclusions: ARMAR hand performance in terms of orientation is better than the performance of Barrett hand; the performance in terms of position of the virtual fingers is similar; the biggest errors appear in the Large Diameter grasp, for the reasons stated before.

The next experiment tests the robustness with respect to the object position errors. We introduced an error of $\rho = 5cm$ in 6 different directions:

$$\begin{aligned}\widehat{p_o} &= p_o + \rho[\cos\beta, \sin\beta, 0] \\ \beta &= \{0, \frac{\pi}{3}, 2\frac{\pi}{3}, 3\frac{\pi}{3}, 4\frac{\pi}{3}, 5\frac{\pi}{3}\}\end{aligned}$$

The difficulty of the problem should be noted: the size of the objects in their biggest axis is around 10cm, so the error is large compared to their size. The results show that the correspondence error increases substantially in the presence of perception errors. This seems natural if we think about the performance of humans grasping object without any tactile or visual feedback. In Chapter 5 we will describe how this error was improved by the usage of tactile feedback, and how visual feedback could be used to improved further more the precision of robot movements.

4.2.2 Real experiments with KUKA arm and Barrett hand

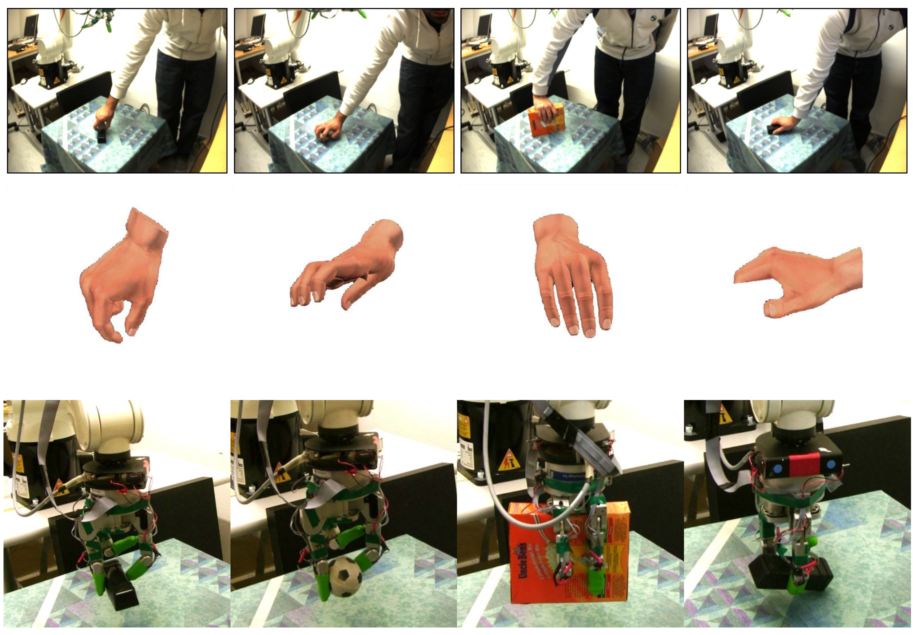

(a) 9 $\rightarrow$ Barrett Precision. (b) 12 $\rightarrow$ Barrett Precision Disc. (c) 1 $\rightarrow$ Barrett Wrap. (d) 4 $\rightarrow$ Barrett Precision.

Figure 4.7: Execution of grasps in a real robot environment: original images, nearest neighbors in the database and robot execution.

Our experimental setup works as follows. An image of the human grasp is captured and passed to the grasp recognition module [8], which returns the type of grasp and the position and orientation of the hand. Using the object pose, the grasp policy is selected and executed. The scenario, illumination and subject is different to the experiments in [8], but we obtain similar results in the classification. Large diameter, small diameter and abducted thumb are correctly classified most of the time, while pinch grasp, power sphere and precision disc grasp are sometimes confused with the power grasp. In terms of orientation, the typical error is around 15 degrees, which is acceptable in the execution of the grasp. The object position is given manually, with an error of ± 3 cm. The position error did not inflict on the grasp execution, except when performing Precision Disc grasp with a ball, which rolled when the hand was not centered over the ball. Fig. 4.7 shows the robot being shown four different grasps (Large Diameter, Abducted Thumb, Pinch and Precision Disc, respectively), mapping them and performing the corresponding grasp (Barrett Wrap, Barrett Precision, Barrett Precision and Barrett Precision Disc, respectively).

4.2.3 Real experiments with ARMAR-IIIb

In this subsection we show how the techniques described in this Chapter were applied towards enabling a humanoid robot, ARMAR-IIIb, to grasp objects based on human demonstration. This system was developed in collaboration with the Humanoids group from the Karlsruhe Institute of Technology (KIT). It integrates modules already existing in KIT such as object recognition [27], upper body tracking [29] and mapping of actions [28] with our grasp recognition and mapping system. We will describe the changes performed on the hand pose estimation system to adapt it to ARMAR-IIIb, and briefly describe the whole system for completeness. Please refer to [9] for more details on the system.

The experimental platform used in this experiment, ARMAR-IIIb, is a copy of the humanoid robot ARMAR-IIIa [25]. From the kinematics point of view, the robot consists of seven subsystems: head, left and right arm, left and right hand, torso, and a mobile platform. The head has seven DoF and is equipped with two eyes, which have a common tilt and can pan independently. Each eye is equipped with two digital color cameras, one with a wide-angle lens for peripheral vision and one with a narrow-angle lens for foveal vision. The upper body of the robot provides 33 DoF: $2 \cdot 7$ DoF for the arms and three DoF for the torso. The arms are designed in an anthropomorphic way: three DoF for each shoulder, two DoF in each elbow and two DoF in each wrist. Each arm is equipped with a five-fingered hand with eight DoF. The locomotion of the robot is realized using a wheel-based holonomic platform.

The system works as follows. First, using a stereo camera setup human observation is initiated by capturing upper body motion and scanning the scene for known objects to attain information on the approach stage. When the human hand is in the vicinity of the object, hand pose estimation is triggered and provides grasp classification and hand orientation. Finally, the motion data is gathered and mapped onto the robot for execution. The mapping is accomplished via a standardized interface and the ensuing execution is achieved by means of non-linear optimization.

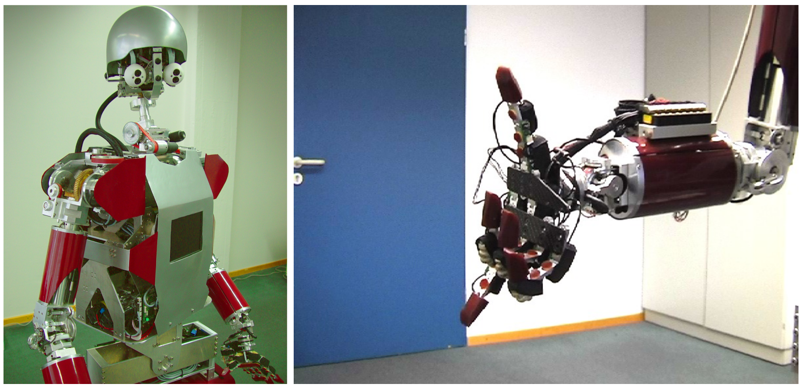


Figure 4.8: The humanoid robot ARMAR-IIIb.

4.2.3.1 Changes in Hand Pose Estimation Module

.

The hand pose estimation module used in this grasping by imitation setup was updated and adapted with respect to the one described in Section 4.1.1. The three main differences are the following:

- **Temporal likelihood.** As explained in Section 2.4 and [3], modeling the temporal likelihood improves the quality of the hand pose estimation system substantially. For this application we modeled the temporal likelihood with a single Gaussian centered in the previous best estimation of the hand pose, as described in [3].
- **ARMAR subset of possible robot grasps.** The grasping capabilities of ARMAR-IIIb determined the set of grasps to be recognized by our hand pose estimation system. Two sets of experiments were performed: in the first one, the whole system (grasp observation, mapping and execution) was tested with a reduced set of grasps :power grasp from top, power grasp from side and pinch grasp(see Figure 4.11); in the second one, the set of grasps was extended to five of them (power sphere, prismatic wrap, parallel extension, tripod and pinch) but the execution of the grasp was reduced to the hand pose, keeping the arm still(see Figure 4.12).
- **Grasp classification triggered by object proximity.** In the previous experiment, the grasp classification had to be triggered manually. Here, the distance between the recognized object and the teacher's hand is measured and the grasp is recognized in a set of frames where the object is sufficiently close to the hand. The grasp class estimation G_t is obtained through a majority voting process within the N_p poses with the highest weight ω_j (for our experiments $N_p = 15$). G_t is then smoothed temporally taking the majority vote in a temporal window of N_f frames ($N_f = 10$ in our experiments). This can be seen in Figure 4.9.

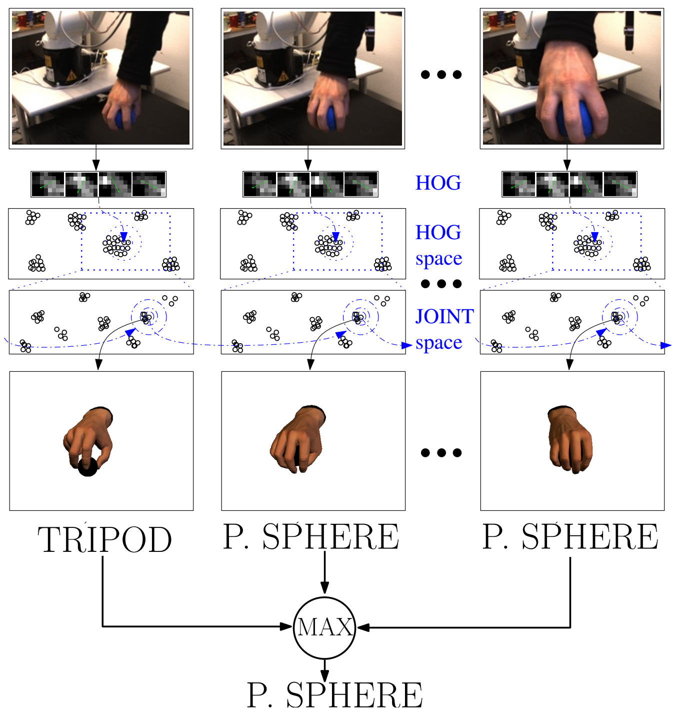


Figure 4.9: Grasp classification with temporal likelihood modeled in joint space

4.2.3.2 Object Recognition

A database-based object recognition system described in [27] was used. The system matches region candidates (obtained through color segmentation) against a database of object views. The information collected from the database is refined through alignment of the model with the real image.

4.2.3.3 Markerless Upper-body Motion Capture

The upper-body tracking system is based on [29]. The system uses a particle filter to estimate a model of the upper body consisting in 14 degrees of freedom. The similarity of model and reality is evaluated based on the similarity of edges and the error in positioning the head and hands of the model [30].

4.2.3.4 Imitation system

The mapping of the demonstration is performed within the Master Motor Map (MMM) framework [28], a standardized interface which combines input from various systems (such as the upper-body tracker and the hand pose estimation) into a common framework.

The grasp type and the estimated hand orientation are passed from the hand pose estimation system to the robot through the MMM interface. Based on this data, the corresponding ARMAR-IIIb grasp is selected. Implementing a grasp type is accomplished by defining adequate joint angles which determine its form. To complete the grasp mapping, the grasp to be performed is adjusted regarding the extent of the object shape. For this purpose a rudimentary grasp type adjustment is implemented, which uses forward kinematics to determine the supposed thumb position on the object shape. The object size is scaled to approximately fit the positions of the remaining finger tips whereas the resulting scaling factor is used to tune the distances between the thumb and fingers.

4.2.3.5 Grasp Execution

The grasp reproduction of ARMAR-IIIb is performed in three different stages. The first stage describes the approach movement of the end effector towards the object based on the observed movement, while in the second stage the end effector is placed to the final grasp pose. The reproduction concludes with the execution of the recognized grasp type.

The reproduction of the arm movements follows the methodology described in [70] where a correspondence error composed by the sum of mean square errors of both wrist position and joint angles is minimized using Levenberg-Marquardt minimization.

For the execution of the final grasp phase, due to the displacement error and further inaccuracies originating from the object localization and the robot's mechanical elements

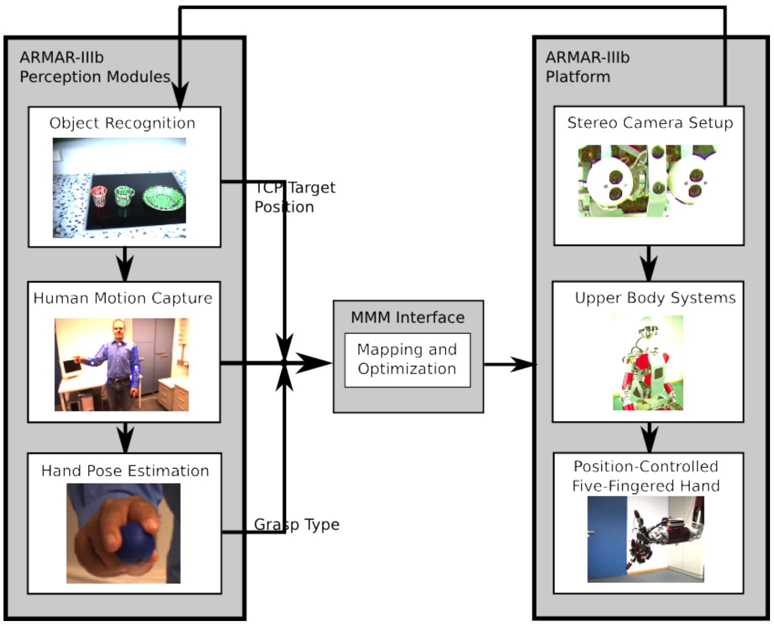


Figure 4.10: Structure of the entire imitation framework.

error arises between the wrist and the object has to be diminished. To achieve exact alignment of the end effector and the robot, we make use of visual servoing methods as presented in [211]. Within this approach the hand and object are tracked. The resulting distance between both is reduced and the hand orientation is controlled. The hand orientation estimate coming from the grasp recognition module is used to determine if the grasp should be executed from the top or from the side. Therefore, the hand is placed over the object if the palm orientation was similar to the table plane, or next to the object otherwise.

4.2.3.6 Experimental Setup

For the experiments, easily identifiable objects, such as cups, were used. The human stands in front of the robot and demonstrates the grasp. As mentioned before, the robot observes the upper body of the human and the applied grasp type and the robots notices the cup which was grasped by evaluating the position of the cups on the table. The robot looks for the cup to be grasped on the table and grasps the same object in the same way. The hand pose recognition system was running in a external computer, while the rest of the system was running in the computer inside of ARMAR-IIIb. It is possible to run the whole system on the robot, but this setup was more preferable for debugging purposes. As mentioned previously, two sets of experiments were performed. The first one has three grasp types to be identified power grasp from top, power grasp from side and pinch grasp, 4.11). The second one has a richer set of five grasps (power sphere, prismatic wrap, parallel extension, tripod and pinch) but the arm mapping was not performed (only the hand mapping was executed, see Figure 4.12).

4.2.3.7 Experimental Results

As depicted in Figures 4.11 and 4.12 the robot successfully imitated the demonstrated grasp including approach and grasp type. Videos of those experiments are available [1] [2] Since a non-linear optimization method is applied during approaching, we attained a tradeoff between a high similarity of the demonstrated movement concerning the reproduction and a unique solution in terms of joint angles, which is not possible using standard IK methods due to singularities and redundancies. Furthermore, the goal-directedness could be retained in task space featuring a smaller displacement error of the end effector towards the object. Concerning the approach phase, we experienced a displacement error of max 65 mm. The displacement could be recovered by using visual servoing, which enables us to visually adjust the hand position with respect to the object. In order to test the grasp classification module, each grasp was executed 20 times for the Experiment 2. The results are shown in Table 4.1. An overall classification accuracy of 72% was achieved, clearly over the human baseline for grasp recognition with similar grasps [8], with four out of five grasp types with accuracies over 80%. The differences between the human and synthetic model had a stronger effect in the parallel extension grasp, resulting in a lower accuracy for that particular grasp.

[1] `http://www.csc.kth.se/~jrgn/GraspRecognitionDivx.avi`
[2] `http://www.csc.kth.se/~jrgn/GraspRecognition_Part2.avi`

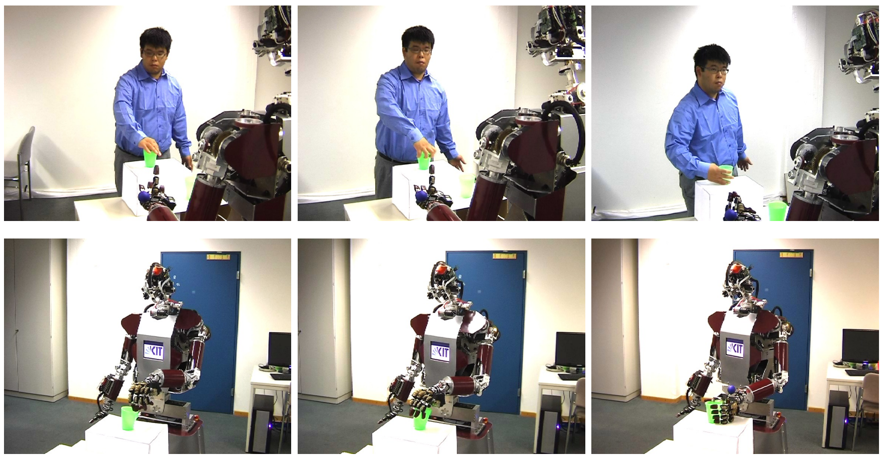


Figure 4.11: Image samples of the online imitation of human motion by the humanoid ARMAR-IIIb; Experiment 1

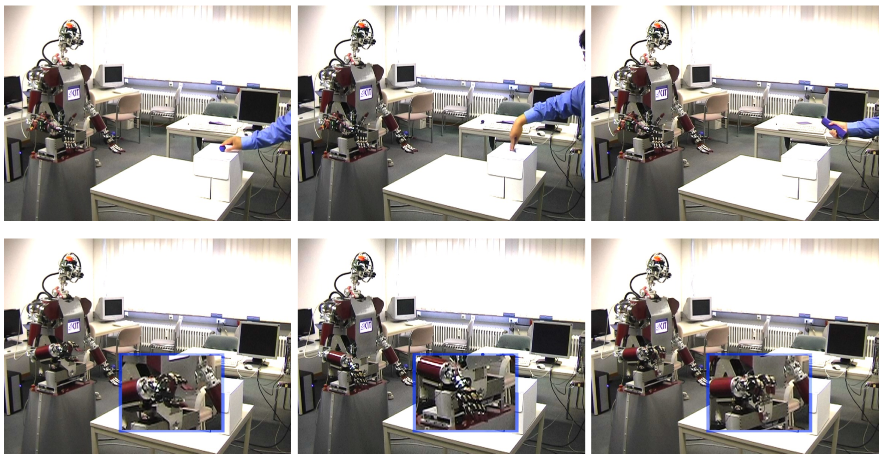


Figure 4.12: Image samples of the online imitation of human motion by the humanoid ARMAR-IIIb, Experiment 2

4.3 Summary

In this Chapter we have presented a system that maps human grasps to robot grasps. The mapping is based on the classification of the teacher grasp into different grasp classes which have a correspondent robot grasping action. We proposed a correspondence measure which can evaluate similarity of grasping poses for hands with different number of fingers. Such a correspondence measure was not applicable in our testing scenario, but its usage can be

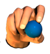
Grasp type	Power Sphere	Prismatic Wrap	Parallel extension	Tripod 	Pinch
Class. rate	80%	95%	50%	85%	80%

Table 4.1: Grasp classification rate for the grasping-by-demonstration system implemented on ARMAR III-b.

useful in simulation for assessing the validity of a different mapping strategy. The approach was demonstrated in three different setups: simulation, a robotic setup with a KUKA arm and a Barrett hand, and on the humanoid robot ARMAR.

4.4 Discussion and future work

The work presented in this Chapter represents a first step towards the mapping of grasps from a human teacher to a robot learner. Although it is been shown to be already usable on a humanoid robot, there are a number of aspects that can be improved in our setup. The remainder of this Section explains three of such improvable aspects.

4.4.1 Closing the execution loop

It is unrealistic to expect that grasping in a real scenario can be planned beforehand, in a linear perceive→plan→execute way. The main reason for this is the presence of inaccuracies in the perception step, which cannot be corrected until the grasp is executed. Such inaccuracies include errors in the estimation of the teacher's hand pose, errors in the estimation of the object pose, robot calibration errors, etc. A second reason for the inadequacy of *open-loop* grasping is the dynamism of realistic grasping scenarios; the object might have changed its pose between the perception and execution phase, the robot might be moving while it tries to grasp the object, etc. For these reasons we consider crucial to *close the loop* in grasp imitation, i.e. include a perception loop that checks if the task is being executed according to our plans, and corrects if that is not the case.

The visual servoing approach used in our ARMAR experiment [211] is an instance of such perception loop: through visual input, the position of the robot wrist is brought to the intended goal with respect to the object. However, the actual pose of the fingers is unknown, since the only tracked feature is a sphere in the robots wrist and the fingers do not provide feedback about their pose (proprioception). This aspect will be discussed in more depth in Chapter 5.

4.4.2 Learning how to imitate

In this chapter we mapped grasps to robots by defining a set of equivalent grasp poses. These equivalences were manually defined. A remaining question is if these equivalent classes could be learnt automatically.

To the best of our knowledge, the main attempt to map grasp demonstrations to a generic manipulator without manually defined equivalence classes is [119]. However, the perceptive information required by this approach (contact points and normals) is difficult to obtain in a real scenario, and, it likely to be very sensitive to noise. Moreover, some information might be missing when a grasp is characterized only by the contact forces. For example, the pose of the palm can be relevant when the purpose of the grasp is to hand the object over to another subject.

Therefore we think the correspondence measure defined in [119] should be modified in two ways. First, it should take into account higher level concepts such as percent of object covered surface, for example. This would account for the absence of palm's pose in the correspondence measure, avoiding at the same time problems such as the lack of a palm in some robotic hands. Second, the measure should be able to deal with uncertain or missing information. The ability in a Bayesian Network framework of marginalizing over missing data [186] seems promising in this aspect.

Given a correspondence measure applicable to real scenarios, the robot could optimize its pose in terms of this correspondence metric.

4.4.3 Imitating sequences of actions

The current system only performs imitation of a single grasp. Unless lifting an object is the main goal of the system (e.g. a robot lifting the carpet in order to allow cleaning under it), this system should be integrated with other imitation modules in order to perform a complete task. One of the main difficulties in such systems is parsing the demonstration into subtasks that can be processed efficiently [97]. However, these subtasks are not independent since all of them have a common goal. Therefore the system should infer subgoals (suitable for this particular robot) which can lead to the final goal. It is likely that those subgoals do not exactly correspond to the world state after each of the subtasks is executed by the teacher.

Chapter 5

Grasp Execution

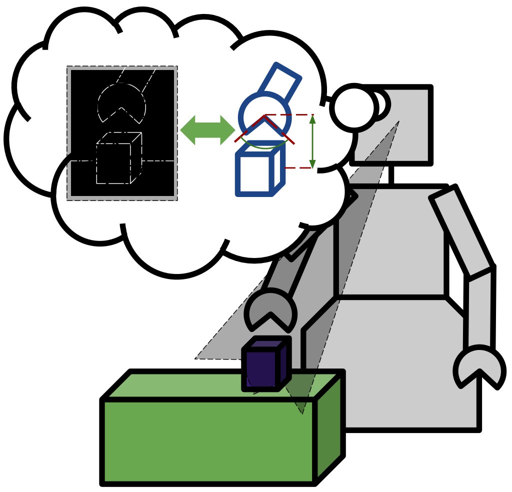


Figure 5.1: Once the robot has a plan for the grasp execution in terms of motor commands, it should observe that the perceived state of the world (represented in the figure as an edge map of the gripper and the object) corresponds to the predicted state of the world, and correct the plan if not.

In previous chapters we have studied how to plan grasps based on human demonstrations: Chapter 2 showed how to estimate the hand pose of a human teacher, Chapter 3 described how to model grasping actions and in Chapter 4 we mapped the human hand poses to robotic grasps. A number of systems, e.g. [51, 178], consider that the imitation problem finishes here, with the solutions to the record mapping $g_R(z,a)$ and the embodiment mapping $g_E(z,a)$ (solved in our case in chapters 2 and 4).

The solution of those problems provide the learner with the means of reproducing the task with his own embodiment. If we consider the case of, for example, a child learning how to tie their shoelaces, the equivalent procedure would consist of observing an adult teacher performing the task, planning how to do it (observing his own hands and shoes), and finally trying to do it with his eyes closed and ignoring the tactile input coming from his fingers. This is possible, but there are several difficulties the child can encounter in such learning task. First, its perception of the world, in this case the shoelaces, might be inaccurate to the level of detail required by the task. These inaccuracies cannot be corrected if the child discards all sensorial input gathered while the task is executed. Second, the outcome of its actions might not be exactly as he expected. The proprioception capabilities of the human body are limited, and therefore our pose might not correspond to our mental model of it when we are deprived of other senses. Moreover, the interaction with objects might have unexpected outcomes. Third, the world state (the pose of the child's shoelaces in our example) might have changed after we have planned our actions due to external agents .

All these difficulties are reasons for continuous usage of perception in human activities. Various experiments have shown that humans re-target their reaching movements without penalty in execution time when the target is moved [93] and when the perception of their own embodiment is visually distorted [113]. When the changes are small (up to 10% of the distance to the object) this adaptation occurs without involving the ventral path of the brain, in charge of higher level tasks such as object recognition, and without people consciously recognizing those adaptive movements. Goodale et at. show in [94] that a delay of two seconds between perception and execution triggers a grasping strategy based on the object information provided by the ventral stream, which results on a slower execution with fixed grasping aperture, less tailored to the details about the object that the dorsal stream provides online.

This intuition translates well into robots, since robots will encounter similar (if not harder) difficulties when they execute grasps "blindly". First, robotic perception is, in general, more inaccurate than human for high level tasks such as estimating hand or object poses. Second, robotics planning relies on robot calibration, the equivalent to human proprioception, and this calibration can be erroneous and/or degrade with time. Finally, the world state changes due to external factors affect robots and humans in the same way.

Our goal in this chapter is to provide tools that complement the systems described in chapters 2 to 4 by facilitating the creation of an "inner simulator" that reflects accurately the state of our body in relation with the world. This "inner simulator" connects with the simulation-theory interpretation of mirror neurons [91] and with the inner simulators used in robotics [69]. However, in both cases those simulators were predicting only the evolution of the external world, while in our case this simulator should also predict and correct the state of our own embodiment. Essentially, this represents an instance of Shadowing Imitation (Figure 1.2) in which the physical robot imitates his internal model where the action is planned, by making the perception from both systems match.

In robotics (and more concretely in robotics grasping) there are two main branches of research following this path. First, the usage of local sensory input such as tactile sensors for correcting action plans, called *corrective movements* or *reactive movements*. Second, the usage of visual data for robust trajectory planning towards a goal, called *visual* *servoing*. In Section 5.1 we describe how the performance of the mapping explained in Chapter 4 improves after the implementation of a simple corrective movements approach. In Section 5.2 we show how matching a visual simulation of our robotic arm and the visual input from our cameras helps us correcting the calibration errors inherent to our system.

5.1 Corrective movements for grasp imitation

As it has been said previously, predicting the outcome of our action enables us to detect unexpected outcomes. When a grasping action is started, object and manipulator are two separate entities; during the grasp, the manipulator comes into contact with the object, sustaining such contact for the duration of the grasp. The robot can predict, based on its initial position, object position and movement plan, the contact time between manipulator and object, together with the contact point in the manipulator. It seems reasonable to detect such transition in order to check that the plan is being executed as intended. This transition occurs in the close proximity of the robot manipulator, and therefore does not require remote sensing such as vision, which usually needs higher level processing.

One of the first systems that provided correction of grasping strategies based on local sensing is described in [197]. An array of LED emitters and receivers inbuilt in a parallel jaw gripper provides information about where the object is. Patterns of reception/no-reception are interpreted and mapped into actions which should improve the object grasping. A more modern system based on proximity optical sensors in shown in [107], where sensor data is used to ensure that all the fingers touch the object at the same time.

Although these type of sensors offer a convenient way of non-intrusive local perception, it is much more common to find manipulators with integrated tactile sensing based on actual touch, [104]. Hsiao et al. [105, 106] have recently provided algorithms to use tactile data for the goal of robust grasping.

The system described in [105] makes use of tactile sensors in three different phases of the grasping action. First, in “Reactive approach”, the approach to the object is corrected depending on where the first contact is detected. Second, in “Compliant closing”, the tactile sensing data is used to make sure that both fingers enter in contact with the object. Finally, in “Grasp adjustment” the tactile arrays ensure that the contact area is centered in the inner part of the fingertip. Another system which divides the grasping task into stages and corrects the execution according to tactile input can be found in [83]. They use both arrays of pressure sensors on the fingertips and force torque sensors on the wrist. The usage of the force torque sensor allows them to detect force normals which are used to align the hand and object (“Alignment Phase”). In a second phase, the forces exerted by the fingers are controlled by sensing the pressure on the fingertips to make them parallel and stable (“Parallel face detection”). Finally, the fingertip forces are increased until they reach a predefined limit (“Force adaptation”).

The choice of manipulator trajectory performed in the “Reactive approach” in [105] is designed as a set of heuristics based on which tactile sensor senses contact first. This re-planning of the trajectory can be performed in a probabilistic manner as a Partially Observable Markov Decision Process (POMDP), [116]. This is done in [106], where the belief state about the object surface position of the world is updated after contact is sensed, and next actions can be chosen depending on it. This results in the robot executing actions which main goal is to gather information about the exact location of the object.

5.1.1 Approach and Evaluation

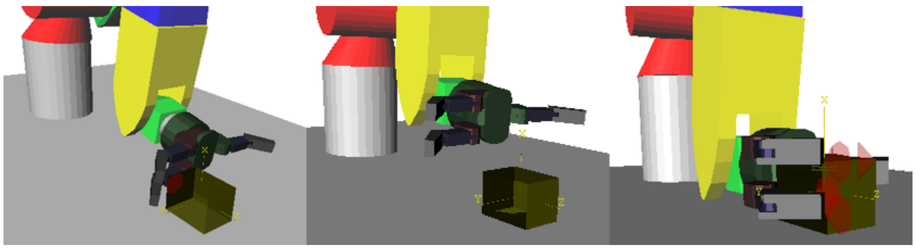


Figure 5.2: Our robot tries to grasp an object, but detects an unexpected contact (represented by its contact cone in red) on a finger. Then it backs up (middle figure) and re-executes the grasp, until the first contact occurs on the palm as initially expected.

In this section our goal is to improve the robustness to positioning error obtained in Section 4.2.1. For this purpose we implemented a simple corrective approach which re-plans the transport phase of the grasp when an unexpected contact is sensed. The “Reactive approach” phase from [105] follows a similar methodology.

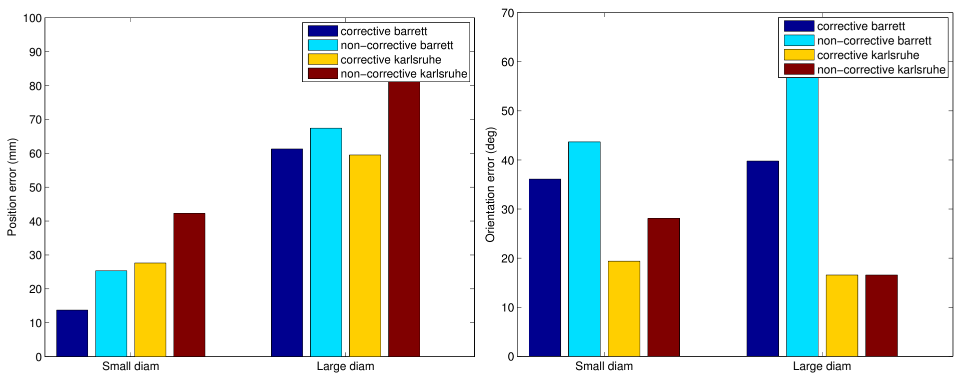
The execution of the volar grasps expects an initial contact palm-object, followed by wrapping the fingers around the object. It is easy to detect that there is a problem in the execution when the first contact occurs on the fingers, see Figure 5.2. If this is the case, the robot backs up and re-plans the grasp. The new goal position for the hand is a weighted average between the detected contact p_c and the original goal position $p_{h\to o}$, $p_{h\to o} = \alpha p_c + (1-\alpha)p_{h\to o}$. We repeat this sequence of grasp re-planning until the first contact of the approach movement occurs on the palm. The corrective modifications of grasp mapping algorithm can be seen in Algorithm 2. The virtual finger error is reduced both for the position and orientation error, as can be seen in 5.3.

In the non-volar grasps (Pinch and Precision grasps, Figure 4.4) the first contact to sense occur directly on the inner part of the fingers. An erroneous object pose estimation will usually result in the first contact occurring on the outer part of the fingers. Since our physical Barrett hand has only sensors on the palm and inner part of the fingers, such contacts are not detected and we do not have a corrective movements approach for the non-volar grasp. This results on a lower error robustness in such grasps, as can be seen in the previous chapter, Figure 4.6.

This experiment shows us principally the importance of the feedback in the presence of errors. It can be seen that, in the presence of object pose errors, the error increases much more in the grasps without corrective movements (grasps 3, 4, 5 and 6) than in the ones with corrective movements (grasps 1 and 2). Another fact that can be inferred from Fig. 4.6 is that the ARMAR hand is more sensitive to the errors that the Barrett hand; the error

```
/* grasp recognition                                          */
/* grasp mapping                                              */
Set hand to preshape $G_r$;
Set hand to orientation $o_{h \to o}$;
if $G_h \in \{LargeDiameter, SmallDiameter\}$ then ;          /* if volar */
    Approach following $a$ towards $p_{h \to o}$ until contact;
    while contact $p_c$ out of palm do
        Retreat;
        $p_{h \to o} = \alpha p_c + (1 - \alpha) p_{h \to o}$;
        Approach following $a$ towards $p_{h \to o}$;
        $\alpha = \alpha^2$;
    end
else                                                          /* if non-volar */
    Approach following $a$ towards $p_{h \to o} - \delta$;
end
Grasp;
```

Algorithm 2: Pseudo-code for grasp mapping with corrective movements.



(a) Position error (b) Orientation error

Figure 5.3: Error in position (mm) and orientation (degrees) for the two grasps tested with corrective movements (Small Diameter and Large Diameter). The different bars represent the error done by barrett hand with and without corrective movements, and karlsruhe hand with and without corrective movements.

increases more for ARMAR hand in presence of errors than for the Barrett hand. Actually the error for the ARMAR hand in non-corrected grasps is higher than the one showed, because the thumb usually fails to touch the object, and therefore the thumb virtual finger is not compared. There are principally two reason for this worse robustness to errors: first, the shorter length of the fingers; second, the palm configuration in the ARMAR hand. The shorter length of ARMAR fingers affects the non-volar grasps, as we can see in Fig. 5.4. The small distance between the base of the thumb and the base of the rest of the fingers affects the volar grasps, that usually collide with the finger bases before touching the palm. However, this is mostly solved by the corrective movements.

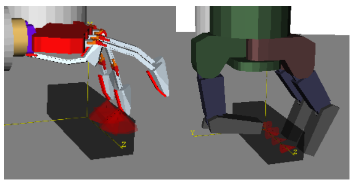

(a) (b)

Figure 5.4: Example of performance of a grasp without tactile or visual feedback.

5.2 Virtual Visual Servoing

In the previous section we have studied how tactile information can be used to ensure that the grasp execution is evolving as it was planned. The main advantage of tactile information is that it provides reliable information which requires little processing for detecting contact between the manipulator and the object. However, relying on tactile information only has some drawbacks as well. First, tactile information is inherently local. In practice this means that we can only detect deviations from the planned grasp once we enter contact with the object. This means that large portions of the grasping action have to be re-executed when errors are detected, while those re-executions might be avoided by correcting the trajectory continuously during the grasp approach. Second, the grasp planning might rely on global information which is not accessible through tactile feedback. For example, if the robot is learning how to grasp a cylinder for handing it to the human, the grasp should be executed on one extreme of the object. However, the global information about where the fingertips are touching the object is not accessible through tactile information. Finally, as noted in the previous section, the most common type of local sensors on the manipulators are intrusive, which means it is likely that they will change the object state after sensing it. In this case, the object pose should be re-estimated and the grasp re-planned.

For all these reasons we propose a system which keeps track of the scene visually. More specifically, in this section we present a system which allows us to track our robot arm and hand to correct a critical component in our grasping system: the position and orientation of the manipulator with respect to the camera (Figure 5.6). An accurate estimation of these variables is critical for the task of grasping unknown objects. Forward models based on offline calibration of the system do not provide sufficient accuracy because of the low repeatability of the motors in the active head [13](see Figure 5.5). The methodology presented in this section is applicable to any entity in the scene for which we have an initial pose estimation and a CAD model, so it could be applied for example to track the pose of the object or the configuration of compliant body parts without proprioception.

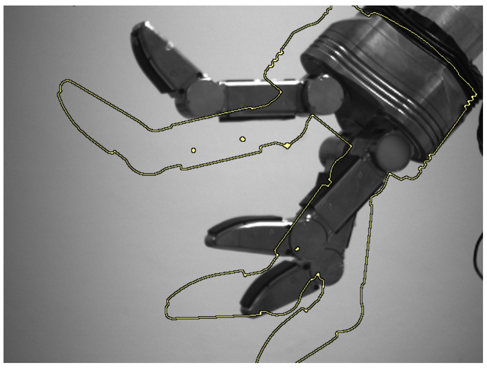


Figure 5.5: The pose of the robotic manipulator, according to the forward model coming from the calibrated kinematics, is overlaid in form of white contour on top of the real image.

In humans, the reaching component of grasping is concerned with bringing the hand to the object to be grasped [114]. In robots this is usually implemented as an approach phase in which the robot manipulator is brought to the vicinity of the object. In order to perform the rest of the grasping action, the position and orientation of the manipulator should be known accurately. A rough estimation of the pose can be obtained from the kinematics of the manipulator and the camera-robot calibration. However, sometimes a kinematic model may not be accurate or available at all.

Classically, vision based control — *visual servoing* [123] has been used to align the robot hand with the object prior to applying a grasp to it. The most common solution to this problem is to place fiducial markers on the robotic manipulator. For cases in which the manipulator has a big planar surface, ART tags [121] are a common choice, see Figure 5.7a. However, more dexterous manipulators resembling the human hand may not have a surface to which such a marker can be attached. In these cases one possibility is to rigidly attach an easily trackable object to the wrist, like the red sphere in Figure 5.7b. This marker allows us to infer the relative position of the wrist with respect to the camera, [211]. However, the perception of the red sphere cannot provide any information about the orientation of the wrist (Figure 5.8, left column). These markers are rarely situated on the fingertips in order to avoid self-occlusion; this means that the flexion of the fingers cannot be controlled visually (Figure 5.8, middle column). This can represent a problem for compliant hands such as ARMAR-IIIb or Barrett hand. Although it can be alleviated by the inclusion of position sensors [36], this introduces another calibration stage prone to introduce errors in the system. Another problem with this approach is that the mobility of the manipulator gets effectively reduced, because it should keep the marker always visible (Figure 5.8, right column).

In order to alleviate the limitations described above, we propose an approach that con-

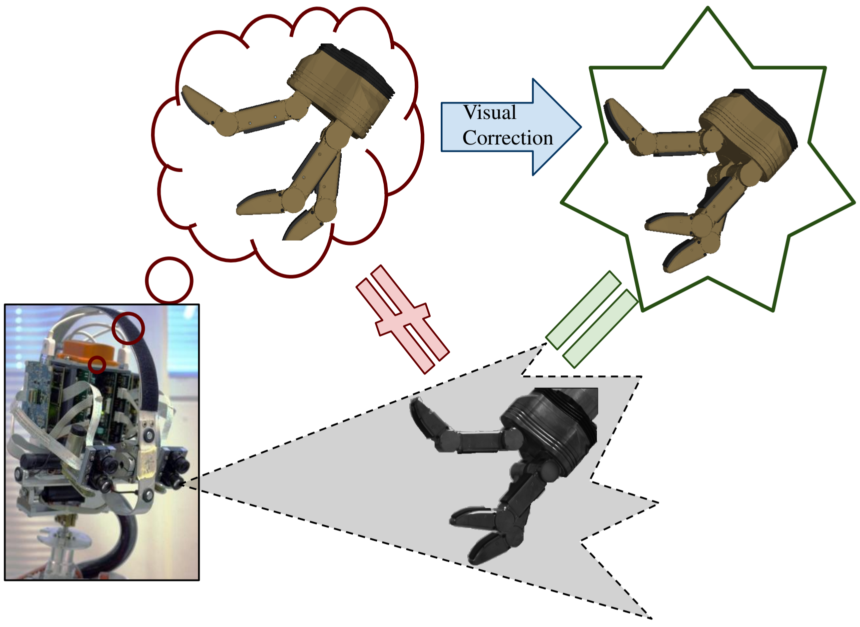


Figure 5.6: The real image from the cameras (bottom) is compared with the forward model coming from proprioceptive sensors (top left). Since they do not match, the model is corrected so that they match visually.

sists in tracking the manipulator itself instead of a marker placed on it. By tracking the manipulator, we obtain an accurate estimation of its pose and we do not rely on the visibility of the markers.

Different approaches for tracking complex objects, potentially articulated, have been discussed in Chapter 2. There we concluded that, given the real-time requirements of the application and the high dimensionality of the human hand, a discriminative approach was the best solution. The problem that we consider in this chapter shares the same goal, but diverges in terms of its characteristics. First, an approximation of the pose of the robotic manipulator is available through calibration, while there is no initialization available in the human hand pose estimation. This makes the usage of local tracking methods feasible. Second, in this chapter we will estimate the pose of our robot-hand system, concerning six degrees of freedom, as opposed to the minimum twenty degrees of freedom of the human hand articulation (plus six degrees of freedom of general position and orientation). Third, the complexity of the human hand makes very difficult the creation of models visually accurate; on the other hand, our robotic arm-hand model fits very well the real images obtained.

For all these reasons we decided to track the robot using a generative approach. There is a number of systems performing accurate, real-time tracking of rigid structures through

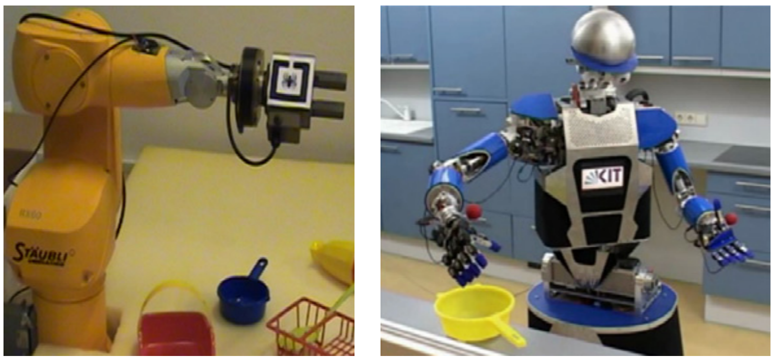

(a) ART marker (b) Red spherical markers

Figure 5.7: Different markers on robotic hands. Images extracted from [165].

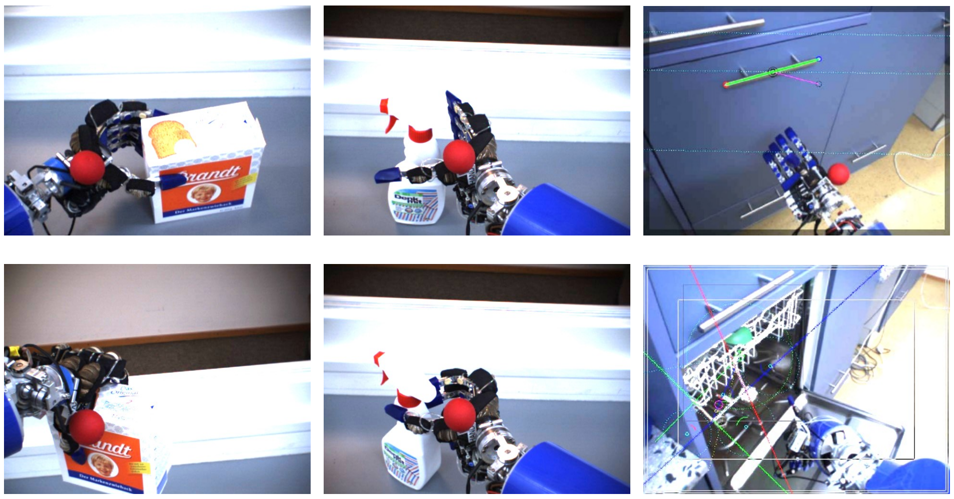


Figure 5.8: The spherical marker does not provide information about the wrist position (figures on the left side) or about the fingers flexion (figures on the middle). Those parameters are therefore not controllable visually. In the right column, after opening the dish-washer door the red marker gets occluded by the robot arm .The images from the left-most column are extracted from [211]; the rest from [109].

a generative approach, [73, 60]. The principle driving all these systems is similar, as it is noted in the comparison [59]. They have a model of the appearance of the object to be tracked. By minimizing the visual mismatch between model and reality, the pose error is minimized. This minimization is done by changing the pose of the model so that the visual error decreases. The visual error in both of them is based on the distance from a subset of contour points from the model visualization to the closest real edge lying on the normal to the contour. There are two main differences between those systems. First, in [73] the sampling of the contour is done in 3D, while in [60] the sampling is done in 2D. Second, the minimization problem is solved iteratively for each frame in [60], while it is non-iterative for [73]. For more details about the comparison between those methods, please refer to [59].

The system described in this chapter differs from the ones described above mainly in two points. First, the models used in our approach are common CAD models which describe the robot's surface, instead of mathematical descriptions of the model contours. This is particularly different in [60], where the features used require the model to be defined in terms of lines and ellipses. Our method is therefore applicable to a wider range of robots without requiring any pre-processing of the model. Second, the visual error between model and real image is based on the Chamfer distance [46] between real and model edge sets. The Chamfer distance does not restrict matching edges to be on the normal to the model contour. The computation of the Chamfer distance is performed efficiently by computing the distance transform [41] of the edges extracted from the real image.

5.2.1 Methodology

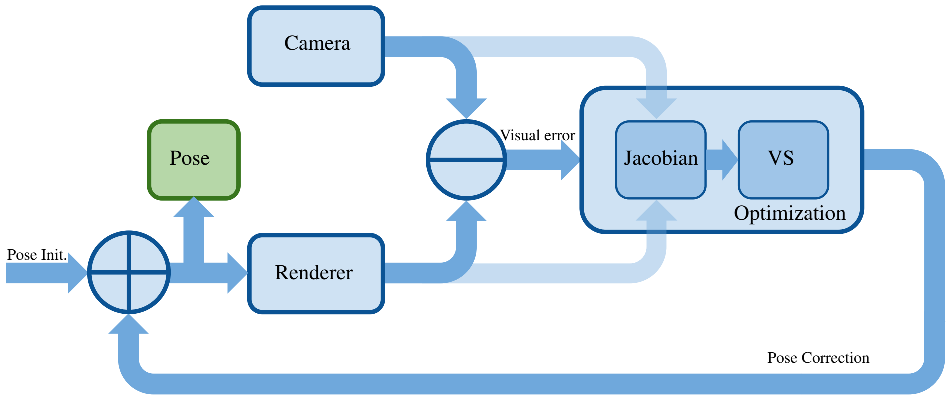


Figure 5.9: Outline of the proposed model-based tracking system.

The pose of an object is denoted by $\mathbf{X}(\mathbf{R}, \mathbf{t})$ where $\mathbf{t} \in \mathcal{R}(3)$, $\mathbf{R} \in \mathbf{SO}(3)$. The set of all poses that the robot's end–effector can attain is denoted with $\mathcal{T}_G \subseteq \mathbf{SE}(3) = \mathcal{R}(3) \times \mathbf{SO}(3)$. A positioning task in general can then be represented by a function

$$\mathbf{e} : \mathcal{S} \rightarrow \mathcal{R}(n) \tag{5.1}$$

where $\mathcal{S}$, in the case of position based visual servoing, may represent the task-space of the end–effector, $\mathcal{T}_G$. In this case, $\mathbf{e}$ is referred to as the *kinematic error function*, [110]. The dimension n of $\mathcal{R}$ in (Eq. 5.1) depends on the number of degrees of freedom (DOF) of the robot constrained by **e** with $n \leq m$ (m denotes DOF of the manipulator). In the case of image based visual servoing, $\mathcal{S}$ represents *image feature parameter space*, $\mathcal{F}$ and $n \leq k$ where k is the dimension of $\mathcal{F}$.

Pose estimation considers a computation of a rotation matrix (orientation) and a translation vector of the object (position), $\mathbf{X}(\mathbf{R}, \mathbf{t})$:

$$\mathbf{X} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & T_X \\ r_{21} & r_{22} & r_{23} & T_Y \\ r_{31} & r_{32} & r_{33} & T_Z \end{bmatrix} \tag{5.2}$$

The equations used to describe the projection of a three–dimensional model point **Q** into homogeneous coordinates of the image point $[x\ y]^T$ are:

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \mathbf{R}\,[\mathbf{Q} - \mathbf{t}] \quad \text{with} \quad \begin{bmatrix} wx \\ wy \\ w \end{bmatrix} = \mathbf{P} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \tag{5.3}$$

where **P** is a 3x3 matrix representing the internal camera parameters matrix including focal length and aspect ratio of the pixels, w is the scaling factor, **R** and **t** represent the rotation matrix and translation vector, respectively.

Our approach to pose estimation and tracking is based on virtual visual servoing where a rendered model of the hand is aligned with the current image of the hand. The outline of the system is presented in Fig. 5.9. In order to achieve the alignment, we can either control the position of the object to bring it to the desired pose or move the *virtual* camera so that the image perceived by the camera corresponds to the current camera image, denoted as *real* camera image in the rest of the paper. In this paper, we adopt the first approach where we render synthetic images by incrementally changing the pose of the tracked object. In the first iteration, the position is given based on the forward kinematics. Then, we extract visual features from the rendered image, and compare them with the features extracted from the current camera image. The details about the features that are extracted are given in the next section.

Based on the extracted features, we define a difference or an error vector between the desired values for the features and the current ones. Based on this error vector, we can estimate the incremental change in pose in each iteration following the classical image based servoing control loop. This process continues until the difference vector is smaller than a certain threshold. Each of the steps is explained in more detail in the following subsections. Our current implementation and experimental evaluation is performed for the KUKA R850 arm and Schunk Dexterous hand, shown in Fig. 5.10.

5.2.2 Virtual image generation

As we mentioned before, we use a realistic 3D model of the robotic parts as input for our system. This adds the challenge of having to render this model at a high frame rate, since our system runs in real time, and several visual servoing iterations must be performed for every frame that we obtain from the cameras.

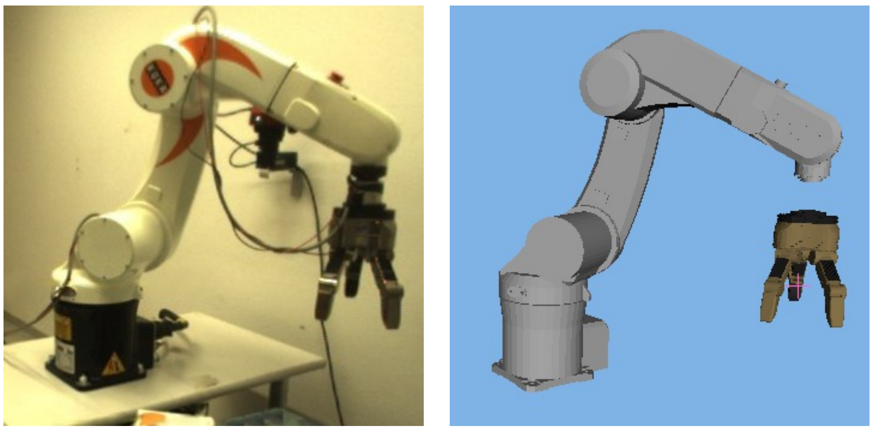


Figure 5.10: The Kuka R850 arm and Schunk Dexterous Hand in real images and CAD models.

To render the image, we use a projection matrix $\boldsymbol{P}$, which corresponds to the internal parameters of the real camera, and a modelview matrix $\boldsymbol{M}$, which consists of a rotation matrix and a translation vector. The modelview matrix is then estimated in the visual servoing loop.

One of the most common CAD formats for objects such as robotic hands are Inventor files. There are a number of rendering engines which can deal with such models either natively or through the usage of plugins, such as Ogre [152] or OpenSceneGraph [154]. These rendering engines are designed and optimized to show the model appearance on the screen. However, we are interested in a very different task: obtaining the 3D points of the model surface directly in memory for further processing. This kind of rendering was slow, overcomplicated or even impossible in the rendering engines tested.

For this reason we developed a new scenegraph engine, specific for this application, which focused on rendering offscreen images at a high speed. It was developed directly over OpenGL. We can obtain about 1000 frames per second with this engine. At the moment the speed of the rendering engine does not represent a bottleneck to our system.

We render the image without any texture or lighting, since we are only interested in the external edges of the model. We also save the depth map produced by the rendering process, which will be useful later for the estimation of the jacobian.

5.2.2.1 Features

The virtual and the real image will be compared in terms of visual features. The choice of those features is crucial because it can affect drastically the speed and the reliability of the system. The most important characteristics of such a feature are the following:

- Speed: The feature will be computed once per frame on the real image and once per iteration (several per frame) on the virtual image. Therefore the feature computation could become the system's performance bottleneck. Since the virtual image features are computed much more often, features with an asymmetric computational time can be beneficial.

- Robustness towards illumination changes: Our robotic hand model, as many other models, does not have a proper model for the specularity and reflectance of the hand surface. Consequently, the selected feature should not depend on the specular reflections or the reflected color on the hand surface. The main characteristic that the feature should reflect is shape.
- Directed error: Some formulations of the error computation in visual servoing depend on the direction which minimizes the error. Therefore, it is preferable to have features where the error direction can be extracted, so that it is possible to know how this error will change with respect to motion of the object.
- Robustness for non-textured models: Robot manipulators are usually plain colored or metallic; therefore features which rely on texture should be avoided.

Harris corners [101] or SIFT [132] are features which can be extracted fast and reliably. However, they depend heavily on texture and are therefore not suitable for our application. In [61] the distance between point and a line, and between ellipse and a line is used. Those features by themselves do not provide a directed error. But more importantly, they are not suitable for complex models with curved lines. Moreover, the creation of the virtual model is either manual or involves detecting linear or ellipsoidal patches, which may be non-trivial depending on the complexity of the model.

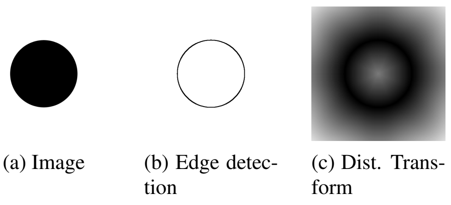
The Chamfer distance [46] can be considered as a generalisation of the features used in [61]. Instead of measuring distances between shapes and points, it directly measures the distance between points and their closest match. The sets of points $\mathcal{U}, \mathcal{V}$ considered are usually edges extracted with a Sobel or Canny operator from images U, V (in our case we will consider U the real image and V the image synthesized from the model). While standard chamfer distance measures the distance between two whole shapes (as the average of the point-to-point distances), we will focus on the individual distances:

$$d_{cham}(v, \mathcal{U}) = \min_{u \in \mathcal{U}} \|v - u\|, v \in \mathcal{V} \tag{5.4}$$

This is a good feature candidate because its computation is fast and it is robust to illumination and color changes. In its original design it does not provide directed error, but that characteristic can be easily added with negligible computational cost:

$$o_{cham}(v, \mathcal{U}) = \angle(v - u), v \in \mathcal{V}, u = \arg\min_{u \in \mathcal{U}} \|v - u\| \tag{5.5}$$

We want to compute the similarities between one real image and multiple virtual images for each frame. Therefore it makes sense to precompute the distance from each pixel from the image to the closest edge for the real image.That is the so called distance transform [41]:



(a) Image (b) Edge detection (c) Dist. Transform

Figure 5.11: Image, edge detection and distance transform of a circle

$$d_{transf}(p) = \min_{u \in \mathcal{U}} \|p - u\|, p \in U \tag{5.6}$$

$$d_{cham}(v, \mathcal{U}) = d_{transf}(v), \qquad v \in \mathcal{V}, u \in \mathcal{U} \tag{5.7}$$

$$\tag{5.8}$$

Once we compute d_{transf}, we can compute $d_{cham}(v, \mathcal{U})$ multiple times for different images rather fast ($O(n)$ with the number of edge points). Therefore, we can compute the distance transform d_{transf} of the real image and then check how similar are the virtual images in linear time with the number of edge points of each virtual image. The simplest form of this feature has a problem; any edge, no matter its shape or orientation, will have a low distance value if positioned in a zone with high density of edges. This can be alleviated by matching edges only if they have similar orientation. For that purpose we use the horizontal and vertical edge images $\mathcal{U}_h, \mathcal{U}_v$ to divide the edge image $\mathcal{U}$ into 8 channels of different overlapping orientation ranges (see Figure 5.12 for an example with four channels). This method is similar to the 3D distance transform described in [131], but in our case we want to discriminate, rather than use some weighted metric for edges with different orientations, so we calculate the distance transform over overlapping channels, effectively rejecting edges with a different orientation.

$$o_u = \arctan(\frac{u_v}{u_h}) \pmod{\pi} \tag{5.9}$$

$$o_u \in [\frac{\pi i}{8}, \frac{\pi(i+1)}{8}] \Rightarrow u \in \mathcal{U}_i \tag{5.10}$$

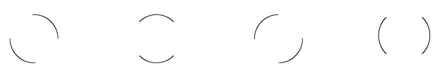
$$d_{transf}(p) = \min_{u \in \mathcal{U}_i} \|p - u\|, p \in I_u \tag{5.11}$$

$$d_{cham}(v, \mathcal{U}) = d_{transf}(v), v \in \mathcal{V}_i, u \in \mathcal{U}_i \tag{5.12}$$

$$s = \{d_{cham}(v, \mathcal{U})\}, d_{cham}(v, \mathcal{U}) < \delta \tag{5.13}$$

This means that edges with different orientations do not affect the distance transform of each other, and therefore they do not get “attracted” by those edges.

Interior edges are more sensitive to illumination changes, shadows, reflections and model imperfections than exterior edges. That is why we decided to extract only the silhouette of the virtual model and try to localize that silhouette in the real image. This



(a) 0° – 90° (b) 45° – 135° (c) 90° – 180° (d) 135° – 225°

Figure 5.12: Edge detection divided into 4 overlapping ranges

avoided the interior edges of the model getting trapped in local minima on the real image, resulting in a more robust matching. It also makes the computation of the chamfer distance faster. However, it can be argued that it makes the virtual model representation ambiguous due to the lack of details in the interior of the silhouette. During our tests this did not represent a problem since the initial position is close enough to the real pose in order to disambiguate between similar silhouettes.

5.2.2.2 Visual servoing

Once the features have been extracted, we can use a classical visual servoing approach to calculate the correction to the pose, [110]. Vision-based control systems can use two different approaches: *Position-based* control uses image data to extract a series of 3D features, and control is performed in the 3D Cartesian space. On the other hand, in *image-based* control, the image features are directly used to control the robot motion. In our case, since the features we are using are distances between edges in an image, for which we have no depth information, we will be using an *image-based* approach.

The basic idea behind visual servoing is to create an error vector which is the difference between the desired and measured values for a series of features, and then map this error directly to robot motion.

Let $s(t)$ be a vector of feature values which are measured in the image. In our case, it is constructed, at each iteration, with the distances between the edges in the real and synthetic images

$$\boldsymbol{s}(t) = \left[d_1, d_2, \ldots, d_n\right]^T \tag{5.14}$$

then $\dot{\boldsymbol{s}}(t)$ will be the rate of change of these distances.

The movement of the manipulator (in this case, the virtual manipulator) can be described by a translational velocity $\boldsymbol{T}(t) = \left[T_x(t), T_y(t), T_z(t)\right]^T$ and a rotational velocity $\boldsymbol{\Omega}(t) = \left[\omega_x(t), \omega_y(t), \omega_z(t)\right]^T$. Together, they form a velocity screw:

$$\dot{\boldsymbol{r}}(t) = \left[T_x, T_y, T_z, \omega_x, \omega_y, \omega_z\right]^T \tag{5.15}$$

We can then define the image jacobian or interaction at a certain instant as $\boldsymbol{J}$ so that:

$$\dot{\boldsymbol{s}} = \boldsymbol{J}\dot{\boldsymbol{r}} \tag{5.16}$$

$$\boldsymbol{J} = \left[\frac{\partial \boldsymbol{s}}{\partial \boldsymbol{r}}\right] = \begin{bmatrix} \frac{\partial d_1}{\partial T_x} & \frac{\partial d_1}{\partial T_y} & \frac{\partial d_1}{\partial T_z} & \frac{\partial d_1}{\partial \omega_x} & \frac{\partial d_1}{\partial \omega_y} & \frac{\partial d_1}{\partial \omega_z} \\ \frac{\partial d_2}{\partial T_x} & \frac{\partial d_2}{\partial T_y} & \frac{\partial d_2}{\partial T_z} & \frac{\partial d_2}{\partial \omega_x} & \frac{\partial d_2}{\partial \omega_y} & \frac{\partial d_2}{\partial \omega_z} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \frac{\partial d_n}{\partial T_x} & \frac{\partial d_n}{\partial T_y} & \frac{\partial d_n}{\partial T_z} & \frac{\partial d_n}{\partial \omega_x} & \frac{\partial d_n}{\partial \omega_y} & \frac{\partial d_n}{\partial \omega_z} \end{bmatrix} \tag{5.17}$$

which relates the motion of the (virtual) manipulator to the variation in the features. The method used to calculate the jacobian is described in detail below.

However, we need to compute $\dot{\boldsymbol{r}}(t)$ given $\dot{\boldsymbol{s}}(t)$ in order to correct our pose estimation.

When $\boldsymbol{J}$ is square and nonsingular, it is invertible, and then $\dot{\boldsymbol{r}} = \boldsymbol{J}^{-1}\dot{\boldsymbol{s}}$. This is not generally the case, so we have to compute a least squares solution, which is given by

$$\dot{\boldsymbol{r}} = \boldsymbol{J}^{+}\dot{\boldsymbol{s}} \tag{5.18}$$

where $\boldsymbol{J}^{+}$ is the pseudoinverse of $\boldsymbol{J}$, which can be calculated as:

$$\boldsymbol{J}^{+} = (\boldsymbol{J}^{T}\boldsymbol{J})^{-1}\boldsymbol{J}^{T}. \tag{5.19}$$

The goal for our task is to have all the edges in our synthetic image match edges in the real image, so the target value for each of the features is 0. Then, we can define the error function as the following, which leads us to the simple proportional control law where K is the gain parameter:

$$\boldsymbol{e}(\boldsymbol{s}) = \boldsymbol{0} - \boldsymbol{s} \tag{5.20}$$

$$\dot{\boldsymbol{r}} = -K\boldsymbol{J}^{+}\boldsymbol{s} \tag{5.21}$$

5.2.2.3 Estimation of the jacobian

To estimate the jacobian, we need to calculate the partial derivatives of the feature values with respect to each of the motion components. When features are the position of points or lines, it is possible to find an analytical solution for the derivatives.

In this case, however, the features are the distances from the edges of the synthetic image to the closest edge in the real image, so we approximate the derivative by calculating how a small change in the relevant direction affects the value of the feature.

As said before, we obtained a depth map while rendering the model. This depth map allow us to obtain the 3D point corresponding to each of the edges. If $D(\boldsymbol{u})$ is the distance to an edge for a projected point $\boldsymbol{u}$ we can calculate the derivative for a model point $\boldsymbol{U}$ with respect to T_x as:

$$\frac{D(\boldsymbol{PM}(\boldsymbol{U} + \epsilon\boldsymbol{x})) - D(\boldsymbol{PMU})}{\epsilon} \tag{5.22}$$

where ϵ is an arbitrary small number and $\boldsymbol{x}$ is a unitary vector in the x direction. A similar process is applied to each of the motion components.

5.2.3 Experimental evaluation

In this section we evaluate our method, both qualitative and quantitatively. We analyse the robustness of our approach with respect to errors in the initial estimation of the robots pose and errors in the visual features extraction. In order to run these qualitative experiments in our real robot we would need pose ground truth data coming from a "perfect" pose estimation system. Since this was not available in our setup, we conduct those experiments in simulation. The real experiments were conducted in the context of a system performing autonomous grasping of unknown objects, [13].

5.2.3.1 Qualitative evaluation

We use the 3D model and rendering engine described in Section 5.2.2 (the same used in the visual servoing loop) to generate manipulator poses with controlled error in terms of initial pose estimation and feature extraction. The process used in these experiments was as follows:

- Generate a rendered image of the model at a known pose $\mathbf{X}(\mathbf{R}, \mathbf{t})$.
- Add some error in translation ($\mathbf{t}_\epsilon$) and rotation ($\mathbf{R}_\epsilon$) to that pose, and use this as the initial pose estimation $\mathbf{X}_\epsilon(\mathbf{R} \times \mathbf{R}_\epsilon, \mathbf{t} + \mathbf{t}_\epsilon)$.
- Run the virtual visual servoing loop with the rendered image as input. In some of the runs, noise was added to the detection of edges, to assess the robustness with respect to certain errors in the edge detection.
- After each iteration, check whether the method has reached a stable point (small correction) and the difference between the detected pose $\mathbf{X'}(\mathbf{R'}, \mathbf{t'})$ and the known pose $\mathbf{X}(\mathbf{R}, \mathbf{t})$ is below a certain threshold. If this is the case, consider it a successful run and store the number of iterations needed.
- If the system does not converge to the known pose after a certain number of iterations, stop the process and count it as a failed run.

We ran three sets of experiments, each of them focusing on the following kind of input error:

- Translational error. We evaluated which range of errors in the translational component of the initial pose estimation allows the system to converge. To that effect, we added, in each run, a translational error of known magnitude and random direction.
- Rotational error. We also evaluated the range of errors in the rotational component of the initial pose estimation for which the system converges to the correct result. For each run, we added a rotational error of known magnitude and random direction.
- Error in edge detection. To simulate the effects of wrongly detected edges in the performance of the system, we added random errors to the edge detection of the input synthetic image. In each run, the detection of a number of pixels was shifted by a random amount. An example of the result can be seen in Fig. 5.13.

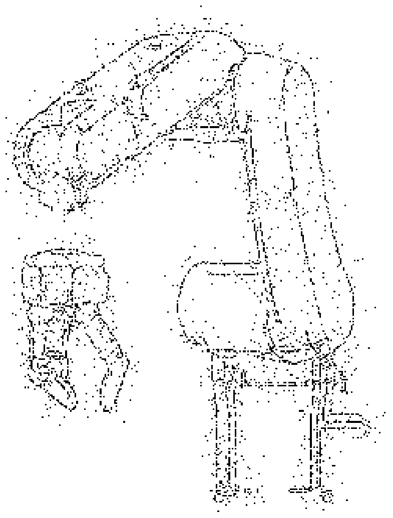


Figure 5.13: Edge detection with artificially introduced error (in 30% of the pixels).

The performance of the VVS system is measured in (i) the number of runs that failed and (ii), for the runs that succeeded, the average number of iterations it took to do so. We plot these with respect to the magnitude of the input error. For each value of the input error, we ran 500 visual servoing loops. In each of these runs, the magnitude of the error was kept constant, but the direction (or the particular pixels that were affected in the case of edge detection) was chosen randomly. Also, the whole process was repeated for five different configurations of the joints of the arm.

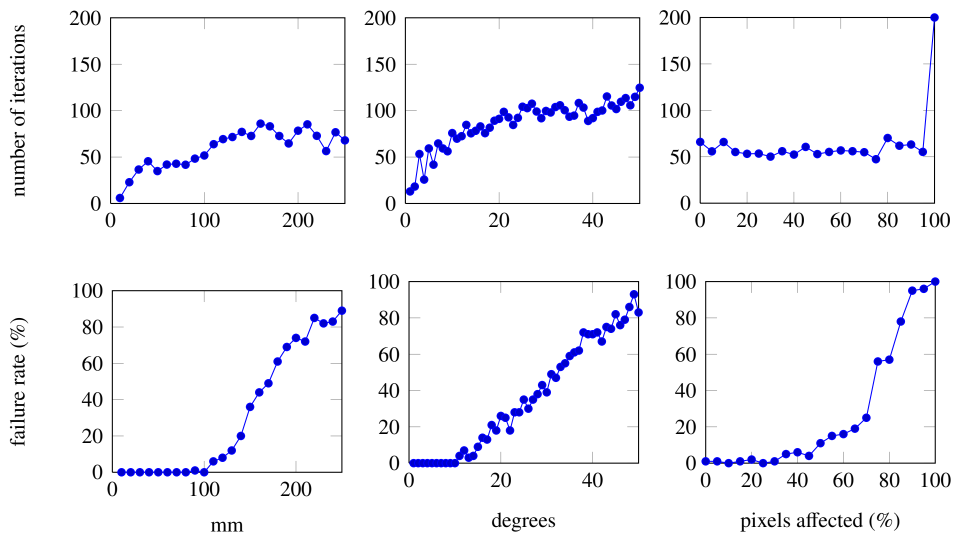
The rest of the parameters of the process were set as follows:

- The threshold for deciding that the algorithm had converged to the correct pose was 10 mm in translation and 0.5° in rotation. This threshold was empirically determined so that it guaranteed real convergence, but was not small enough that there were stability issues.
- The number of iterations after which the run was considered a failure was set to 200.
- For the experiments that evaluate robustness with respect to edge detection, a translation error of 100 mm and a rotation error of 10° was used for the initial pose estimation.

Figure 5.14a shows the result of increasing the translational component of the error in the estimated initial pose. As we can see, the failure rate is close to 0 for distances below 100 mm, and then starts increasing linearly. This is probably due to 100 mm being in the same order of magnitude as the distance between different edges of the robot which have the same orientation, so the system gets easily lost, trying to follow the wrong edges. We can also observe an increase in the iterations needed for reaching the result.

In Figure 5.14b we can observe a similar behaviour for the tolerance to errors in the rotational component of the estimated initial pose. Here, the threshold is around 10° and the reason is probably the same as before: this is the minimum rotation that bring edges to the position of other edges.

With respect to the error in edge detection, we can see in Figure 5.14c that when less than 50% of the pixels are wrongly detected, the system performs almost as well as with no error, and after that the performance quickly degrades. It is also significant that for the runs that converge successfully, the number of iterations is almost independent of the errors in edge detection.



(a) Effects of translational error (b) Effects of rotational error (c) Effects of errors in edge detection

Figure 5.14: Effects of different errors in the performance of our approach. Each column represents an error source: initial translational pose error, initial rotational pose error, and errors in the edge detection (Figure 5.13). Each row shows two measures of how much the error affects the performance: the number of iterations it takes to converge, and the percent of trials which fail to converge. The experiments regarding errors in edge detection were performed with translational error of 100 mm and rotational error of 10°.

5.2.3.2 Qualitative evaluation

For our experiments with the real robot, we set up a table-top scenario with two randomly placed objects, as can be seen in the last picture in Figure 5.15. The head was initially

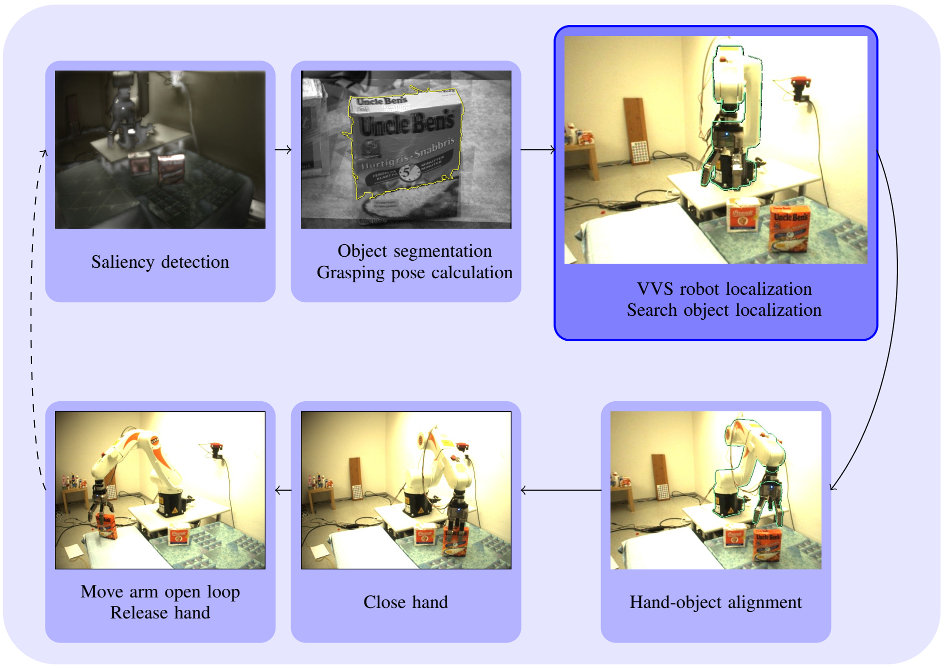


Figure 5.15: Grasping system diagram. The modules represented with small boxes are modules were not implemented by the author of this thesis and therefore are not covered here; more information available in the related publication.A video with the whole sequence can be found at `http://www.youtube.com/watch?v=e-O3Y8_cgPw`

looking towards the table, where it could see the objects and find them in the saliency map of the attention system. However, the arm was not fully visible.

Virtual visual servoing was only running after the object was segmented, because the fixation of the cameras on the object left the arm mostly out of the image view. With this setup the pose estimated through virtual visual servoing quickly converged even if the initial estimation for the pose of the arm was significantly different. In Figure 5.15 we can see, outlined in green, the estimated pose for the arm and hand over the real image.

However, even if they did not affect the performance of the system, there are some problems that arise with the use of virtual visual servoing in real images. For example, as we can see in the upper-right picture of Figure 5.15, the upper edge of the reconstructed outline does not match exactly with the upper edge of the arm. There are two main reasons for this. First, the illumination of the scene can lead to some edges disappearing, because of the low contrast. This is usually not a problem, because the correct matching of other edges will compensate for this. But in this case, there is a second problem: because of the orientation of the arm with respect to the camera, the outer silhouette has really few edges. As we can see in the picture below that, once the pose of the arm changes, the system can recover and show the correct pose again.

5.2.4 Summary

In this section we have presented a system which corrects online the execution of a robotic manipulator based on its visual appearance. Our approach generalizes the visual features and robot models used in previous approaches, [73, 60]. The system improves the accuracy of the manipulator pose estimation obtained through forward model by offline calibration. Although it was tested on an autonomous grasp setup, the approach is equally applicable to a grasping-by-demonstration task. This system represents a first step towards a more complete online simulation of a grasping scenario, in which the pose of compliant joints (such as the ones in the Barrett hand) and objects in the scene are estimated as well. Such a simulation would help to overcome the three problems of open loop execution stated in the introduction to this chapter: inaccurate perception, inaccurate execution and scene dynamism.

5.2.5 Discussion and Future work

The system shows good performance in a realistic robotic setup considering a state-of-the-art robot hand. There are some future directions that can improve the performance of our system and make it more robust. Some of these directions are ongoing work that has not been properly evaluated yet.

5.2.5.1 Denser visual features

The tracking of model edges instead of fiducial markers on the robot represents an improvement on the robustness of the manipulator pose estimation with respect to occlusions. However, there are situations in which even denser features could help our algorithm to avoid local minima. For example, in Figure 5.15 top right, two problematic conditions were met: first, the color contrast between robot and background was rather poor, making difficult the extraction of edges on the top of the robot; second, the robot has a pose with a small number of contour points. The result of these conditions was that the model converged to a pose which shares most of the contour points with the visualization, but fails to match the top edges (undetected). Although this did not affect substantially the performance of our system, we should improve our vision input so that we can handle this situation robustly.

Another improvable aspect of the edge matching is their sensibility to local minima. Our algorithm tries to make each edge in the model's contour match an edge in the image. However, it is possible that the model's contour points are matched to edges in the image which do not belong to the robot if the initial errors displaces the edge towards a cluttered area in the scene. This can lead to convergence to incorrect local minima.

For these two reasons we started considering image depth, from either Kinect or stereo images, to help us improve the robustness of the system. One way to integrate depth in our system is to pre-segment the robot taking into account depth [40], avoiding in this way the erroneous matchings between model and background. Another method consists on extending the 2D matching of edges to 3D matching of surface points, [86]. This approach is more promising, because robust segmentation is still a big challenge in computer vision research. However, matching 3D points can be computationally expensive.

5.2.5.2 Optimization improvements

In our system, all the features with an matching error below a fixed threshold are considered. This represents a crude form of robust estimators. Comport et al. discuss in [61] that estimating the confidence in the features is re-estimated at each iteration improves the results. Therefore, a wider range of robust estimators with online estimation of the confidence should be tested in the future. We are also considering other approaches for the least-squares problem in visual servoing, such as Levenberg-Marquardt [63], which might speed-up convergence.

5.2.5.3 Algorithm initialization

The errors of the forward model of the robot in each frame are clearly not independent. However, our initializes the pose of our model according to the forward model every frame, disregarding the optimized pose obtained in previous frames. It is apparent that integrating the proprioceptive information of the robot, which inform us about its dynamic behaviour, with the previous estimation of the pose, which uncovers the error in the forward model, can provide a much better initialization for our system.

5.2.5.4 Simultaneous tracking of object and manipulator

One important aspect of realistic grasping scenes is not yet covered by our system: object dynamism. There is a number of factors that can cause objects to move during a grasping action: slippage, tactile exploration, initial instability of the object pose, etc. Our method can be easily extended to track an object additionally to the robot manipulator, given that we have a model of the object and an initial pose for it. A number of algorithms can provide us such information with low latency, e.g. [58]. Tracking simultaneously the object and the manipulator would enable us to perform accurate visual servoing of the manipulator relative position with respect to the object, instead of servoing the pose with respect to the camera and then estimating the object pose in camera-frame as well.

Chapter 6

Summary and Discussion

In this thesis we have presented the required components for enabling a robot to learn grasping actions from a human teacher. These components have been divided into four groups: perception, modeling, mapping and execution, which correspond to Chapters 2 to 5.

In Chapter 2 we dealt with the problem of perceiving a grasping demonstration from a human teacher. More specifically, our goal for this problem is to estimate the pose of the teacher's hand in terms of joint angles. To make the demonstration more natural, no markers or external devices should be used to simplify the perception problem. This allows us to potentially learn manual actions from already recorded footage, like videos from the web. In Section 2.3 we tackled the extraction of 3D hand contours for the purpose of fingertip position estimation and, potentially, hand pose estimation. Its lack of robustness made us switch to a different paradigm: discriminative hand pose estimation. By exploiting a database of hand poses with typical object occlusions, we manage to estimate hand poses without markers and under occlusions in real-time. The discriminative nature of the approach imposes a limit in the achievable accuracy; a possible future line of work consists of including a generative optimization phase which can adjust further the results of the discriminative pose estimation. Another way of improving the system resides on the integration of the hand pose estimation with perception modules related to the hand pose, such as arm pose estimation or object pose estimation.

We believe that the performance of our hand pose estimation system can also benefit from learning a better representation of the visual feature space and pose space. *Pose representation* is precisely the concept treated in Chapter 3. There, we try to create two compact models for human grasps in terms of hand pose. This relates to the concept of action synergies, widely used in neurophysiology and robotics. While this concept has always been used in conjunction with linear dimensionality reduction, in this chapter we show how it is not only possible, but also better, to use more modern non-linear dimensionality reduction methods such GP-LVM. In order to motivate our selection of this algorithm we study in depth the assumptions each of the algorithms (PCA and GP-LVM) makes on the data, and how these assumptions affect different aspects of the models, such as reconstruction error and generalization across subjects. The main future direction for this module, apart from further investigating the possibilities of GP-LVM to shape the manifold based on prior information, is to use the knowledge gathered about grasp models in the rest of the modules of our system. In order to do that, the pose space in the hand pose estimation module (joint-space) has to be made compatible with the space in the modeling module (task-space). In order to use the modeling for the execution on the robot, the mapping from human kinematics to robot kinematics should be further studied.

The mapping of kinematics between different embodiments is the main topic covered in Chapter 4. This problem is far from trivial; ideally, the robot should perform a grasp that allows it to perform the same task as the teacher. However, sometimes this is difficult, if not impossible, due to robot's limitations in high level reasoning. For example, if the action to be imitated is to pour water from a jar, the critical part to imitate is to leave the jar hole uncovered to allow for such an action. This high level reasoning is sometimes difficult to handle even by humans: for example, it is difficult to explain why the grasp of a pencil for writing is done the way it is. The solution in these cases is to try to imitate the pose as accurately as possible. However, differences in embodiment make this imitation difficult. In Section 4.2.1 we developed a correspondence metric, based on the concept of Virtual Fingers, which allows a comparison of poses between manipulators with a different number of fingers. However, since such comparison requires information about contact points between manipulator and object, a simpler, classification-based mapping was actually implemented and executed both in simulation and two real setups. One way to improve such a system is to try to encode this high level reasoning about the tasks in terms of higher level perception. For example, features such as "surface covered by the manipulator", "usage of handles or holes", etc can encode information about the ultimate goal of the action. A promising system working in this direction is shown in [186]. Another way of providing more information about what to imitate would be to perform imitation of sequences of actions, in which the ultimate goal can be actually perceived.

One problem of our approach in Chapter 4 is the lack of robustness to errors and changes in the scene. Errors are bound to occur both in the perception side (e.g. object pose) and in the execution side (calibration of the manipulator). Further, real scenes are also dynamic: there might be changes in the scene between its perception and the execution of the grasp. All these lead to the need of perceiving the action while it is executed in order to correct mismatches between our model and reality. In Chapter 5 we present two modes of what is commonly known as "closing the loop": applying corrective movements through tactile sensing in Section 5.1 and tracking the robotic arm and hand through Virtual Visual Servoing in Section 5.2. The basic corrective movements described in this Chapter react to unexpected contacts (contacts on the fingers while the first contact was expected on the palm) by replanning the approach to the object. The application of corrective movements shows an improvement on the correspondence measurement described in 4.2.1. Virtual Visual Servoing (VVS) is used to estimate the real position on the manipulator, which differs from the modeled one due to calibration errors. By correcting our model of reality we can perform the approach to the object accurately. This technique is applicable to the estimation of passive joints in hands like Barrett or objects in the scene, as soon as we have a 3D model of them and an initialization of their pose. This could enable a virtual simulation of the grasping scenario which is accurate and dynamically coherent with the real scene. One of the ways to improve the performance of the actual implementation of the system is including denser features which are not so sensitive to initialization errors, such as depth information.

6.1 Used techniques

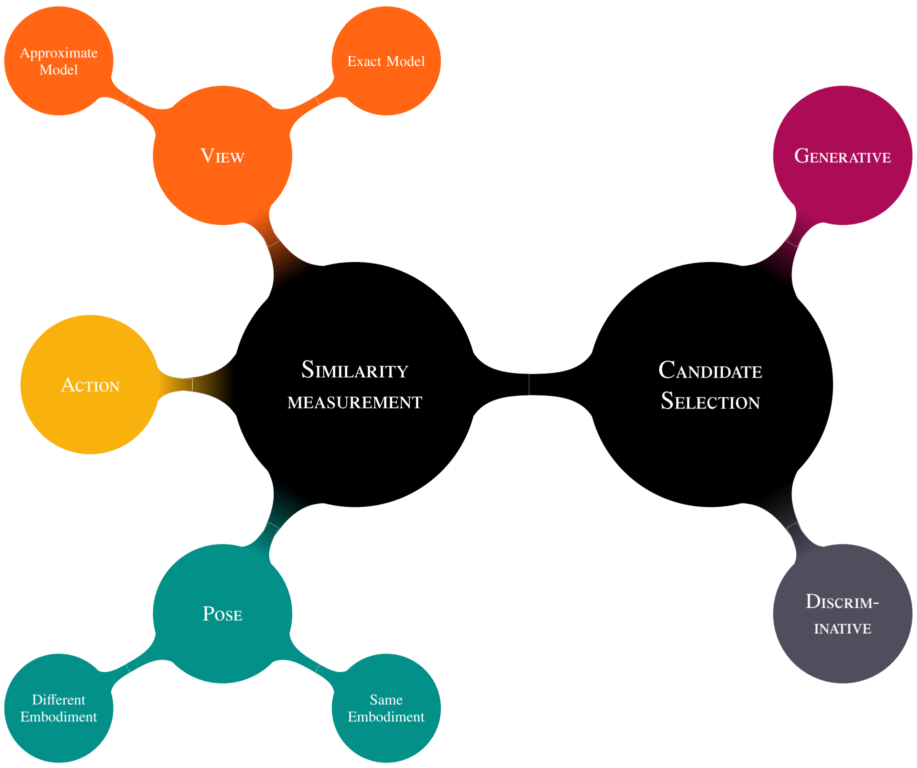


Figure 6.1: Two main concepts have been used in this thesis: similarity measurement and candidate selection. The submodules of our system produce outputs which should resemble their inputs under some constraints; this similarity is defined through different kinds of distance functions. Sometimes the similarity cannot be measured between all possible solutions, so a candidate selection method is triggered to provide promising candidates to the similarity measurement.

The organization of this thesis followed an order related to our application goal, i.e. executing robotic grasps based on human grasping. However, there is a number of similar concepts or techniques that appear in different places in the thesis, which could have shaped the structure of the thesis in a different way. In this section we will provide an overview of these concepts and how they relate to different parts of this document.

6.1.1 Similarity measurement

If it was necessary to choose a single concept related to all parts of this thesis, this would be probably the concept of similarity. The main four chapters of this document describe a process in which an input is transformed into an output, trying to minimize some error measurement. In these processes, our computations try to find an output which is as *similar* as possible to the input in some respect. In order to achieve that, these modules need to specify a distance function through which we can compare the input and possible outputs of the module, and a representation of such elements. A common approach is to represent the elements as vectors of numbers and use the euclidean distance $\mathcal{L}_2$ between these two vectors. However, the choice of $\mathcal{L}_2$ might not be adequate. Also the vectorization of the elements can be done in several ways.

6.1.1.1 Pose

In the problem of robot grasping, and more specifically in a Grasping-by-Demonstration scenario following the "External Observer" paradigm (Figure 1.2), the concept of pose and pose similarity is central. Each of the steps depicted in Figure 1.2a has a pose as an output: human pose is the result of the Human Pose Estimation module, robot pose is the result of Human2Robot mapping, and a refined robot pose is the result of the Grasp Execution.

When the kinematic structures to be compared are similar, two representations widely used in robotics are task-space(spatial coordinates of the end effector(s)) and joint-space (the orientation or joint angle of each of the drivable joints). Joint-space is usually more compact and easily transformable to motor commands in a robot, but can deliver unintuitive similarity measures (Figure 2.21). In Section 2.4 we use joint-space, although task-space representation looks promising for improving the results of that module. In the rest of the thesis task-space is used.

Sometimes it might be useful to incorporate knowledge about the dynamic behavior of poses in time, like when we try to model how likely a human pose is give the pose in the previous frame, Section 2.4.4.2. This can be done by shaping the pose similarity to reflect the dynamic behavior observed in training sequences, see 2.5.1.

The definition of a pose similarity measure becomes more difficult when the two kinematic structures (i.e. embodiments) are significantly different. In Chapter 4 we define a pose representation which is common to two different embodiments. This common representation is related to how the manipulators are actually used for a particular grasp, instead of how raw kinematic configuration is.

A common problem in pose representation and pose control is the high dimensionality of both joint and task-space. In Chapter 3 we focus on the creation of low dimensional pose representations. The shape and characteristics of this space are a result of an optimization problem which incorporates different priors like smoothness or uni-modality (one point in our representation generating a single pose in task-space, see Section 3.2.2.2).

6.1.1.2 View

In the system we envision there are two main connections between the real world and our system. First, the robot records visually the human demonstration of the grasp. Second, the robot corrects its execution by perceiving, visually, the correspondence between its mental model of the world and the reality. Both connections are visual, and consequently complex and of high dimensionality. We should define a compact representation for these images (commonly know as the choice of visual features) and a distance function that allows us to compare them.

Both comparisons in our system will occur between real images and images generated from a synthetic model of the scene (in our cases a synthetic model of the human hand and a synthetic model of our robot). The fact that those models can be inaccurate influences the representation of the images.

In the case of an accurate model, like in our robot model, the features should capture a very detailed representation of the view. A fast, accurate feature with these characteristics is an edge map (see Section 5.2.2.1). When a synthetic model should cope with small variations in the physical body it represents, like in the human hand model case, features should capture a more broad, less detailed essence of the visual appearance. An example is our interpretation of Histogram of Gradients (HOG) features, see Section 2.4.3.1.

There is an aspect that makes visual comparison different to pose comparison in our system. Since the goal of our Hand Pose Estimation system (Chapter 2) is to estimate the *pose* of the human hand, the visual comparison is performed as the means for comparing poses. In other words, since the pose of the human pose is not observable, we compare visual appearance as an indirect way of comparing poses. This insight suggests that the distance function used in the visual comparison should reflect as much as possible similarity in pose space.

6.1.1.3 Action

Chapter 3 creates models of the pose variation of the hand while executing different grasping actions. These models are based on the execution of a particular grasp by different humans. Therefore, there are two main differences between these models and the representation of pose: first, these models should account for the temporal dimension; second, these models should blend information from different subjects. The multi-subject character of this problem encouraged us to use a probabilistic model based on Gaussian Mixture Models (see Section 3.5). The distance function that allows us to compare different grasps and classify new grasp sequences can be defined probabilistically in terms of how likely is a sequence (or a set of sequences) of hand poses given a particular grasp model.

6.1.2 Candidate Selection

Similarity is implicitly defined through a distance function between two elements. The subsystems in our thesis compare their input to candidate solutions (in the cases where we are looking for solutions like in human hand pose estimation Section 2.4 or Virtual Visual Servoing Section 5.2) or with already chosen solutions (when evaluating a solution, like in the human-to-robot mapping evaluation, Section 4.2.1). When the problem is sufficiently small, the space of candidates can be fully traversed, trivializing the candidate selection. This is the case in the selection of the most similar grasping actions in 3.6.2, where only 31 elements are compared.

However, there are other cases in which the set of solution candidates can be very large, or infinite. Indeed, if we consider a problem like pose estimation with solutions in a continuous space $\mathbb{R}^N$, the cardinality of the possible solutions set is theoretically infinite (practically limited by machine precision). In this framework we can consider two main approaches: discriminative and generative.

6.1.3 Discriminative

Discriminative approaches subdivide the space into a finite set of subspaces. This division is usually data-driven, i.e. optimized according to training data. This process makes finite an infinite set, but its size can still be prohibitive. Hierarchical methods such as kd-trees reduce the set of candidates to be evaluated by early disregarding sets of subspaces that do not look “promising”. Hashing methods like Local Sensitive Hashing simplify the distance evaluation into multiple naïve classifiers or hashes, whose results are later combined. Both methods have been tested in the context of human hand pose estimation, Section 2.4.

6.1.4 Generative

The other main branch of candidate selection methods is composed by generative approaches. Generative approaches *generate* candidates from their continuous state space, instead of *selecting* them from a set of possible solutions. The generation of candidates usually follows promising directions set by derivatives of the evaluation function of different orders (gradient descent is a first order method, Gauss-Newton algorithm is a second order method, etc), sometimes partially randomized for robustness (stochastic optimization). The accuracy of these methods is not limited by the granularity of the subdivision of the state space, but by the exploration time of the space. Their expertise lies in exploring local neighborhoods in not-so-high dimensional problems. This is why we used them in Virtual Visual Servoing, Section 5.2.

6.2 Discussion

Grasping-by-Demonstration has proved to be a challenging problem which requires knowledge and development from very different fields, such as Computer Vision, Machine Learning and Control. Its multidisciplinary character has had a double effect on this thesis. On the one hand, it allowed us to explore (and learn about) very different problems, contributing in many different fields with development that can transcend the particular problem attacked here. On the other, it required solving so many hard problems that the final result is not our pursued robust learning robot. We believe we are substantially closer to that dream than when we started this thesis, but it would be naïve not acknowledging the length and slope of the road ahead of us. Particularly promising is the state of real-time pose estimation, both of human hands, robotic manipulators and known objects. The integration of the knowledge about these poses, which feels near at hand, could provide a powerful tool to robots: a visual simulation of all the relevant entities of the grasping scenario. But having an accurate simulation of the visual aspects of the scene would probably be insufficient. A higher level of scene understanding will be needed to imitate effectively complex actions; in the best of the cases, visual knowledge will allow us to imitate exactly what has been taught, allowing only small deviations from the demonstration. Robust robot learners will require the specification of the action's goal in terms of a representation that is at the same time general enough and manageable by the robot. Connecting with this thesis, the modeling and mapping modules depend heavily on the specification of the action's goal, which was simplified to keep it manageable in our case. The development of a general and manageable knowledge representation remains as one of the biggest challenges in computer science and robotics.

Glossary

affine transformation Non-singular linear transformation followed by a translation, i.e. a combination of translation, rotation and shear. In matrix form, $\vec{y}$ and $\vec{x}$ are related by an affinity if $\vec{y} = A\vec{x} + \vec{b}$ or, $\begin{pmatrix} \vec{y} \\ 1 \end{pmatrix} = \begin{pmatrix} A & \vec{b} \\ \vec{0} & 1 \end{pmatrix} \begin{pmatrix} \vec{x} \\ 1 \end{pmatrix}$. 18, 19

covariance function Given a collection of random variables $f(x)$, the covariance function k expresses the correlation between pairs of those random variables: $k(x_p, x_q) = cov(f(x_p, x_q)) = \mathbb{E}[(f(x_p) - \mathbb{E}[f(x_p)])(f(x_q) - \mathbb{E}[f(x_q)])$. A continuous covariance function $k(x_p, x_q)$ models an infinite mixture of random variables $f(x)$.When evaluated in a finite number of points, it should generate valid covariance matrices (symmetric positive semidefinite matrices). 60, 61

covariance matrix Matrix $K \geq 0$ whose elements K_{ij} represent the covariance between two random variables from a finite set. Its origin can be the instantiation of a continuous covariance function $k(x_p, x_q)$ on a finite set $\{x\}$. 58–63

eigendecomposition A square matrix A with size $N \times N$ can be factorized as $A = PDP$, where P is a matrix whose columns are the eigenvectors, $P = \begin{pmatrix} X_0 & X_1 & \cdots & X_N \end{pmatrix}$, and D is a diagonal matrix with the corresponding eigenvalues, $\mathrm{diag}(\lambda_0, \lambda_1, \cdots, \lambda_N)$. 58, 59

eigenvalue For a square matrix A and a non-zero vector v, the eigenvalue λ corresponding to the eigenvector v satisfies the equation $Av = \lambda v$. Eigenvalues also correspond to the elements in the diagonal matrix D when A is decomposed as $A = PDP$. 59

Frobenius Norm The Frobenius Norm of a matrix A of size $m \times n$ corresponds to the euclidean norm of its flattened version: $||A||_F = \sqrt{\sum_{i=0}^{m} \sum_{j=0}^{n} |a_{ij}|^2}$. 58, 62, 66

generative mapping Mapping that transforms latent space to observed space.. 31, 57–61

image segmentation Partitioning of an image into semantic *segments*. 13, 28, 33, 34, 47, 100, 127

intrinsic representation Inherent, essential representation of data in which correlations between dimensions are removed by representing each real degree-of-freedom as a single dimension. 53, 56, 57, 59, 60

joint-space Space parametrized by the degrees-of-freedom of a robot. 13, 53, 65, 85, 129, 132

kernel function covariance function. 60, 61, 63

kernel matrix covariance matrix. 60, 63, 64, 67, 70, 71, 73

latent space Underlying space, usually of lower dimensionality than the original one, that can regenerate the data it was created from. The intrinsic representation of a space is one of its latent spaces, and it is usually the one dimensionality reduction methods strive to extract. 60, 61, 63, 64, 69, 70, 74–76

matrix rank Maximum number of columns or rows from a matrix A (with size $m{\times}n$ which are linearly independent. If the rank $rk(A)$ is lower than min m, n, some columns/rows are linear combinations of a orthogonal basis of dimensionality $rk(A)$. 58, 59, 62

non-parametric model Model which does not assume the mapping between model and observations have a functional form, e.g. histograms or nearest-neighbor models. 13, 28, 29, 31, 32, 37, 44, 48

pose estimation Determination of an entity's configuration in 3D. Referred to rigid objects, it means computing their 3D position and orientation. For articulated objects, the computation should include the internal degrees-of-freedom as well. 7, 11–13, 16–18, 27–29, 31, 32, 41, 44, 47, 48, 85, 90, 96, 99–101, 110, 112, 114, 117, 122–124, 126, 127, 129, 133, 134

prior In the Bayes theorem, the prior is the term $p(A)$, which is multiplied by the likelihood $p(B|A)$ and regularized to obtain the posterior $p(A|B)$. Conceptually, it is related to expert knowledge existent before the evidence is available. 56, 60–62, 86

proprioception Inner sense of the muscle contraction in our body, which give us information about the relative position of body parts. Can be thought as the sensory feedback mechanism for human motor control, equivalent to the motor encoders in robots. 51, 52, 105, 107, 108, 112

task-space Space parametrized by the 3D position (and optionally the orientation) of a robot's end effectors. 13, 53, 65, 85, 116, 129, 132

Bibliography

[15] MESA Imaging AG. Swiss ranger. http://www.mesa-imaging.ch/, 2011. [Online; accessed 31-August-2011].

[16] A Agarwal and B Triggs. Recovering 3D human pose from monocular images. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, pages 44–58, 2006.

[17] S. Ahmad. A usable real-time 3d hand tracker. In *Signals, Systems and Computers, 1994. 1994 Conference Record of the Twenty-Eighth Asilomar Conference on*, volume 2, pages 1257 –1261 vol.2, oct-2 nov 1994.

[18] J. Aleotti and S. Caselli. Interactive teaching of task-oriented robot grasps. *Robotics and Autonomous Systems*, 58(5):539 – 550, 2010. ISSN 0921-8890.

[19] R. O. Ambrose, H. Aldridge, R. S. Askew, R. R. Burridge, W. Bluethmann M. and Diftler, C. Lovchik, D. Magruder, and F. Rehnmark. Robonaut: Nasa's space humanoid. *Intelligent Systems and their Applications, IEEE*, 15(4):57–63, jul/aug 2000.

[20] M. A. Arbib, T. Iberall, and D. M. Lyons. Coordinated control programs for movements of the hand. In A. W. Goodwin and I. Darian-Smith, editors, *Hand function and the neocortex. Experimental Brain Research Supplemental 10*, pages 111–129, 1985.

[21] B. Argall, S. Chernova, M. M. Veloso, and B. Browning. A survey of robot learning from demonstration. *Robotics and Autonomous Systems*, 57(5):469–483, 2009.

[22] A. A. Argyros and M. I. A. Lourakis. Real time tracking of multiple skin-colored objects with a possibly moving camera. In *ECCV*, volume 3, pages 368–379, 2004.

[23] A. A. Argyros and M. I. A. Lourakis. Binocular hand tracking and reconstruction based on 2d shape matching. In *Pattern Recognition, 2006. ICPR 2006. 18th International Conference on*, volume 1, pages 207 –210, 0-0 2006.

[24] A. A. Argyros and M. I. A. Lourakis. Vision-based interpretation of hand gestures for remote control of a computer mouse. In *Computer Vision in Human-Computer Interaction, ECCV 2006 Workshop on HCI, Graz, Austria, May 13, 2006, Proceedings*, volume 3979, pages 40–51. Springer, 2006.

[25] T. Asfour, K. Regenstein, P. Azad, J. Schroder, A. Bierbaum, N. Vahrenkamp, and R. Dillmann. Armar-iii: An integrated humanoid platform for sensory-motor control. In *Humanoid Robots, 2006 6th IEEE-RAS International Conference on*, pages 169–175, dec. 2006.

[26] V. Athitsos and S. Sclaroff. Estimating 3D hand pose from a cluttered image. In *CVPR*, pages 432–439, 2003.

[27] P. Azad, T. Asfour, and R. Dillmann. Stereo-based 6d object localization for grasping with humanoid robot systems. In *IEEE/RSJ International Conference on Intelligent Robots and Systems(IROS), 2007.*, pages 919–924, 29 2007-Nov. 2 2007.

[28] P. Azad, T. Asfour, and R. Dillmann. Toward an Unified Representation for Imitation of Human Motion on Humanoids. In *IEEE International Conference on Robotics and Automation*, Rome, Italy, April 2007.

[29] P. Azad, T. Asfour, and R. Dillmann. Robust Real-time Stereo-based Markerless Human Motion Capture. In *submitted to International Conference on Humanoid Robots*, Daejeon, Korea, December 2008.

[30] P. Azad, A. Ude, T. Asfour, and R. Dillmann. Stereo-based markerless human motion capture for humanoid robot systems. In *IEEE International Conference on Robotics and Automation*, pages 3951–3956, 2007.

[31] A. O. Balan and M. J. Black. An adaptive appearance model approach for model-based articulated object tracking. In *Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Volume 1*, pages 758–765, Washington, DC, USA, 2006. IEEE Computer Society.

[32] R. Balasubramanian, L. Xu, P. D. Brook, J. R. Smith, and Y. Matsuoka. Human-guided grasp measures improve grasp robustness on physical robot. In *ICRA*, pages 2294–2301. IEEE, 2010.

[33] N. Bernstein. Biodynamics of locomotion. In H. T. A. Whiting, editor, *Human Motor Actions. Bernstein Reassessed.*, chapter III, pages 171–221. North-Holland, 1984.

[34] N. Bernstein. The problem of the interrelation of co-ordination and localization. In H. T. A. Whiting, editor, *Human Motor Actions. Bernstein Reassessed.*, chapter II, pages 77–120. North-Holland, 1984.

[35] P. J. Besl and N. D. McKay. A method for registration of 3-D shapes. *IEEE Transactions on Pattern Analysis and machine Intelligence*, 14(2):239–258, February 1992.

[36] A. Bierbaum, J. Schill, T. Asfour, and R. Dillmann. Force position control for a pneumatic anthropomorphic hand. In *IEEE/RAS International Conference on Humanoid Robots (Humanoids)*, pages 21 – 27, 2009.

[37] A. Billard, Y. Epars, S. Schaal, and G. Cheng. Discovering imitation strategies through categorization of multi-dimensional data. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 3, pages 2398–2403, 2003.

[38] S. Birchfield. Elliptical head tracking using intensity gradients and color histograms. In *CVPR*, pages 232–237. IEEE Computer Society, 1998.

[39] S. Bitzer and S. Vijayakumar. Latent spaces for dynamic movement primitives. In *Humanoids 2009*, 2009.

[40] M. Björkman and D. Kragic. Active 3d segmentation through fixation of previously unseen objects. In *Proceedings BMVC*, pages 119.1–11, 2010. ISBN 1-901725-40-5.

[41] G. Borgefors. Distance transformations in digital images. *Computer Vision, Graphics and Image Processing*, 34:344–371, 1986.

[42] A. M. Borghi. Object concepts and action. In D. Pecher and R.Zwaan, editors, *The Grounding of Cognition: The role of perception and action in memory, language, and thinking*, part 2, pages 8–34. Cambridge University Press, 2005.

[43] M. Bray, E. Koller-Meier, and L. Van Gool. Smart particle filtering for high-dimensional tracking. *Computer Vision and Image Understanding: CVIU*, 106(1): 116–129, April 2007.

[44] B. Buchholz and T. J. Armstrong. A kinematic model of the human hand to evaluate its prehensile capabilities. *Journal of Biomechanics*, 25(2):149 – 162, 1992. ISSN 0021-9290.

[45] T. Bugnyar and L. Huber. Push or pull: an experimental study on imitation in marmosets. *Animal Behaviour*, 54(4):817 – 831, 1997.

[46] M. A. Butt and P. Maragos. Optimum design of chamfer distance transforms. *IEEE Transactions on Image Processing*, 7(10):1477–1484, 1998.

[47] J. M. Cabelguen, C. Bourcier-Lucas, and R. Dubuc. Bimodal locomotion elicited by electrical stimulation of the midbrain in the salamander notophthalmus viridescens. *The Journal of Neuroscience*, 23(6):2434–2439, 2003.

[48] S. Calinon. *Robot Programming by Demonstration: A Probabilistic Approach*. EPFL/CRC Press, 2009.

[49] S. Calinon and A. Billard. Incremental learning of gestures by imitation in a humanoid robot. In *Proceedings of the ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pages 255–262, March 2007.

[50] S. Calinon, A. Billard, and F. Guenter. Discriminative and adaptative imitation in uni-manual and bi-manual tasks. In *Robotics and Autonomous Systems*, volume 54, pages 370–384, 2005.

[51] S. Calinon, F. Guenter, and A. Billard. On learning, representing and generalizing a task in a humanoid robot. *IEEE Transactions on Systems, Man and Cybernetics, Part B*, 37:286–298, 2007.

[52] Sylvain Calinon and Aude Billard. Recognition and reproduction of gestures using a probabilistic framework combining pca, ica and hmm. In *Proceedings of the 22nd international conference on Machine learning*, ICML '05, pages 105–112, New York, NY, USA, 2005. ACM.

[53] J. Canny. A computational approach to edge detection. In *RCV87*, pages 184–203, 1987.

[54] M. Charikar. Similarity estimation techniques from rounding algorithms. In *STOC*, pages 380–388, 2002.

[55] M. T. Ciocarlie and P. K. Allen. Hand posture subspaces for dexterous robotic grasping. *I. J. Robotic Res*, 28(7):851–867, 2009.

[56] M. T. Ciocarlie, C. Goldfeder, and P. K.Allen. Dimensionality reduction for hand-independent dexterous robotic grasping. In *IROS*, pages 3270–3275. IEEE, 2007.

[57] Matei Ciocarlie and Peter Allen. Data-driven optimization for underactuated robotic hands. In *2010 IEEE International Conference on Robotics and Automation (ICRA 2010)*, pages 1292–1299. IEEE, May 2010.

[58] A. Collet, M. Martinez, and S. S. Srinivasa. The moped framework: Object recognition and pose estimation for manipulation. *The International Journal of Robotics Research*, 30(10):1284–1306, 2011.

[59] A. I. Comport, D. Kragic, E. Marchand, and F. Chaumette. Robust real-time visual tracking: Comparison, theoretical analysis and performance evaluation. In *ICRA*, pages 2841 – 2846, apr. 2005.

[60] A. I. Comport, É. Marchand, and F. Chaumette. A real-time tracker for markerless augmented reality. In *ISMAR*, pages 36–45. IEEE Computer Society, 2003.

[61] A. I. Comport, É. Marchand, M. Pressigout, and F. Chaumette. Real-time markerless tracking for augmented reality: The virtual visual servoing framework. *IEEE Trans. Vis. Comput. Graph*, 12(4):615–628, 2006.

[62] M. Cutkosky. On grasp choice, grasp models and the design of hands for manufacturing tasks. *IEEE Transactions on Robotics and Automation*, 5(3):269–279, 1989.

[63] D. Marquardt. An algorithm for least-squares estimation of nonlinear parameters. *SIAM J. Appl. Math.*, 11:431–441, 1963.

[64] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *CVPR*, pages I: 886–893, 2005.

[65] T. Darrell, G. Gordon, M. Harville, and J. Woodfill. Integrated person tracking using stereo, color, and pattern detection. *IJCV*, 37(2):175–185, 2000.

[66] T. E. de Campos and D. W. Murray. Regression-based hand pose estimation from multiple cameras. In *CVPR*, pages 782–789, 2006.

[67] M. de la Gorce, N. Paragios, and D. J. Fleet. Model-based hand tracking with texture, shading and self-occlusions. In *CVPR*, pages 1–8, 2008.

[68] Q. Delamarre and O. D. Faugeras. 3D articulated models and multiview tracking with physical forces. *CVIU*, 81(3):328–357, 2001.

[69] Y. Demiris and G. Hayes. *Imitation as a dual-route process featuring predictive and learning components: a biologically-plausible computational model*, pages 327–362. MIT Press, Cambridge, MA, USA, 2002. ISBN 0-262-04203-7.

[70] M. Do, P. Azad, T. Asfour, and R. Dillmann. Imitation of Human Motion on a Humanoid Robot using Non-Linear Optimization. In *IEEE International Conference on Humanoid Robots*, pages 545–552, Daejeon, Korea, December 2008.

[71] W. Dong, Z. Wang, W. Josephson, M. Charikar, and K. Li. Modeling LSH for performance tuning. In *CIKM*, pages 669–678, 2008.

[72] D. Dragulescu, V. Perdereau, M. Drouin, L. Ungureanu, and K. Menyhardt. 3d active workspace of human hand anatomical model. *BioMedical Engineering OnLine*, 6(1):15, 2007.

[73] T. Drummond and R. Cipolla. Real-time visual tracking of complex structures. *IEEE Trans. PAMI*, 24(7):932–946, 2002.

[74] C. H. Ek, P. Torr, and N. Lawrence. Gaussian process latent variable models for human pose estimation. In *Machine Learning for Multimodal Interaction*, pages 132–143, 2008.

[75] Carl Henrik Ek, Philip H.S. Torr, and Neil D. Lawrence. Gaussian process latent variable models for human pose estimation. In *4th Joint Workshop on Multimodal Interaction and Related Machine Learning Algorithms*, pages 132–143, 2007.

[76] S. Ekvall and D. Kragic. Interactive grasp learning based on human demonstration. In *ICRA*, pages 3519–3524, 2004.

[77] S. Ekvall and D. Kragic. Grasp recognition for programming by demonstration tasks. In *ICRA*, pages 748–753, 2005.

[78] S. Ekvall and D. Kragić. Learning and evaluation of the approach vector for automatic grasp generation and planning. In *IEEE International Conference on Robotics and Automation*, pages 782–789, 2007.

[79] S. El Khoury, A. Sahbani, and V. Perdereau. Learning the natural grasping component of an unknown object. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'07)*, pages 2957–2962, San Diego, CA, USA, 2007.

[80] G. ElKoura and K. Singh. Handrix: animating the human hand. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation*, SCA '03, pages 110–119. Eurographics Association, 2003. ISBN 1-58113-659-5.

[81] A. Erol, G. N. Bebis, M. Nicolescu, R. D. Boyle, and X. Twombly. Vision-based hand pose estimation: A review. *CVIU*, 108:52–73, 2007.

[82] Eyetronics. Shape snatcher. `http://www.eyetronics.com/`, 2011. [Online; accessed 31-August-2011].

[83] J. Felip and A. Morales. Robust sensor-based grasp primitive for a three-finger robot hand. In *IEEE/RSJ International Conference on Intelligent Robots and Systems(IROS)*, pages 1811–1816. IEEE, 2009.

[84] R. Feris, M. Turk, R. Raskar, K. Tan, and G. Ohashi. Exploiting depth discontinuities for vision-based fingerspelling recognition. *Computer Vision and Pattern Recognition Workshop*, 10:155, 2004. ISSN 1063-6919.

[85] C. Ferrari and J. Canny. Planning optimal grasps. In *IEEE International Conference on Robotics and Automation*, pages 2290–2295, 1992.

[86] N. Fioraio and K. Konolige. Realtime visual and point cloud slam. In *RSS 2011 Workshop on RGB-D: Advanced Reasoning with Depth Cameras*, Los Angeles, California, July 2011. RSS.

[87] T. Flash and B. Hochner. Motor primitives in vertebrates and invertebrates. *Current Opinion in Neurobiology*, 15(6):660 – 666, 2005. ISSN 0959-4388.

[88] W. T. Freeman and M. Roth. Orientational histograms for hand gesture recognition. In *IEEE International Conference on Automatic Face and Gesture Recognition*, 1995.

[89] J. Friedman and T. Flash. Task-dependent selection of grasp kinematics and stiffness in human object manipulation. *Cortex*, 43(3):444 – 460, 2007.

[90] M. Fuchs, C. Borst, P .R. Giordano, A. Baumann, E. Krämer, J. Langwald, R. Gruber, N. Seitz, G. Plank, K. Kunze, R. Burger, F. Schmidt, T. Wimböck, and Gerd H. Rollin' justin - design considerations and realization of a mobile platform for a humanoid upper body. In *ICRA*, pages 4131–4137. IEEE, 2009.

[91] V. Gallese and M. A. Goodale. Mirror neurons and the simulation theory of mind-reading. *Trends in Cognitive Sciences*, 2(12):493–501, December 1998.

[92] D. J. Giurintano, A. M. Hollister, W. L. Buford, D. E. Thompson, and L. M. Myers. A virtual five-link model of the thumb. *Medical Engineering and Physics*, 17(4): 297 – 303, 1995.

[93] M. A. Goodale. *Modularity in visuomotor control: From input to output*, pages 262–285. Ablex Publishing, Norwood, NJ, 1988.

[94] M. A. Goodale, L. S. Jakobson, and J. M. Keillor. Differences in the visual control of pantomimed and natural grasping movements. *Neuropsychologia*, 32(10):1159 – 1178, 1994. ISSN 0028-3932.

[95] D. Gouaillier, V. Hugel, P. Blazevic, C. Kilner, J. Monceaux, P. Lafourcade, B. Marnier, J. Serre, and B. Maisonnier. The nao humanoid: a combination of performance and affordability. *CoRR*, abs/0807.3223, 2008.

[96] C. Granville, D. Southerland, J. Platt, and A. H. Fagg. Grasping affordances: Learning to connect vission to hand action. In Gaurav Sukhatme, editor, *The Path to Autonomous Robots*, pages 1–22. Springer US, 2009. ISBN 978-0-387-85774-9.

[97] J. Gray, C. Breazeal, M. Berlin, A. Brooks, and J. Lieberman. Action parsing and goal inference using self as simulator. In *Robot and Human Interactive Communication, 2005. ROMAN 2005. IEEE International Workshop on*, pages 202 – 209, aug. 2005.

[98] I. V. Grinyagin, E. V. Biryukova, and M. A. Maier. Kinematic and dynamic synergies of human precision-grip movements. *Journal of Neurophysiology*, 94(4): 2284–2294, 2005.

[99] H. Y. Guan, C. S. Chua, and Y. K. Ho. Hand posture estimation from 2D monocular image. In *3DIM99*, pages 424–429, 1999. URL `http://dx.doi.org/10.1109/IM.1999.805373`.

[100] H. Hamer, K. Schindler, E. Koller-Meier, and L. Van Gool. Tracking a hand manipulating an object. In *ICCV*, 2009.

[101] C. Harris and M. Stephens. A combined corner and edge detector. In *Alvey Vision Conf*, pages 189–192, 1988.

[102] L. M. Herman. *Vocal, social, and self imitation by bottlenosed dolphins*, pages 82–127. MIT Press, Cambridge, MA, USA, 2002. ISBN 0-262-04203-7.

[103] D. Herzog, A. Ude, and V. Kruger. Motion imitation and recognition using parametric hidden markov models. In *Humanoid Robots, 2008. Humanoids 2008. 8th IEEE-RAS International Conference on*, pages 339 –346, dec. 2008.

[104] R. D. Howe. Tactile sensing and control of robotic manipulation. *Advanced Robotics*, 8(3):245–261, 1993.

[105] K. Hsiao, S. Chitta, M. T. Ciocarlie, and E. G. Jones. Contact-reactive grasping of objects with partial shape information. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1228–1235. IEEE, 2010. ISBN 978-1-4244-6674-0.

[106] K. Hsiao, L. P. Kaelbling, and T. Lozano-Pérez. Robust grasping under object pose uncertainty. *Auton. Robots*, 31(2-3):253–268, 2011.

[107] K. Hsiao, P. Nangeroni, M. Huber, A. Saxena, and A. Y. Ng. Reactive grasping using optical proximity sensors. In *IEEE International Conference on Robotics and Automation*, pages 2098–2105. IEEE, 2009.

[108] M. K. Hu. Visual pattern recognition by moment invariants. *IEEE Trans. Information Theory*, 8(2):179–187, February 1962.

[109] K. Huebner, K. Welke, M. Przybylski, N. Vahrenkamp, T. Asfour, D. Kragic, and R. Dillmann. Grasping known objects with humanoid robots: A box-based approach. In *International Conference on Advanced Robotics (ICAR)*, pages 1–6, 2009.

[110] S. Hutchinson, G.D. Hager, and P.I. Corke. A tutorial on visual servo control. *IEEE Transactions on Robotics and Automation*, 12(5):651–670, 1996.

[111] A. J. Ijspeert, J. Nakanishi, and S. Schaal. Learning attractor landscapes for learning motor primitives. pages 1523–1530. MIT Press, 2002. ISBN 0-262-02550-7. URL `http://books.nips.cc/papers/files/nips15/CN01.pdf`.

[112] A. J. Ijspeert, J. Nakanishi, and S. Schaal. Movement imitation with nonlinear dynamical systems in humanoid robots. *Proceedings- IEEE International Conference on Robotics and Automation*, 2002.

[113] L. S. Jakobson and M. A. Goodale. Trajectories of reaches to prismatically-displaced targets: evidence for automatic visuomotor recalibration. *Experimental Brain Research*, 78:575–587, 1989. ISSN 0014-4819.

[114] M. Jeannerod. Intersegmental coordination during reaching at natural visual objects. *Attention & Performance*, IX, 1981.

[115] I. T. Jolliffe. *Principal Components Analysis*. Springer-Verlag, 1986.

[116] L. P. Kaebling, M. L. Littman, and A. R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101(1–2):99–134, 1998.

[117] A. Kanaujia, C. Sminchisescu, and D. N. Metaxas. Semi-supervised hierarchical models for 3d human pose reconstruction. In *CVPR*, 2007.

[118] S. B. Kang and K. Ikeuchi. A framework for recognizing grasps. In *CMU Robotics Institute*, 1991. URL `ftp://reports.adm.cs.cmu.edu/usr/anon/robotics/CMU-RI-TR-91-24.ps.Z`.

[119] S. B. Kang and K. Ikeuchi. Toward automatic robot instruction from perception: Mapping human grasps to manipulator grasps. *IEEE Transactions on Robotics and Automation*, 13(1):81–95, 1997.

[120] S. Bing. Kang. *Robot Instruction by Human Demonstration*. PhD thesis, Carnegie Mellon University, 2009.

[121] H. Kato and M. Billinghurst. Developing AR applications with ARToolkit. In *ISMAR*, page 305, 2004.

[122] C. Kim, D. Kim, and Y. Oh. Adaption of Human Motion Capture Data to Humanoid Robots for Motion Imitation using Optimization. *Integrated Computer-Aided Engineering*, 13(4):377–389, 2006.

[123] D. Kragic and H. I. Christensen. Cue integration for visual servoing. *IEEE Transactions on Robotics and Automation*, 17(1):18–27, February 2001.

[124] V. Kruger, D. Herzog, S. Baby, A. Ude, and D. Kragic. Learning actions from observations. *Robotics Automation Magazine, IEEE*, 17(2):30 –43, june 2010.

[125] J. J. Kuch and T. S. Huang. Vision based hand modeling and tracking for virtual teleconferencing and telecollaboration. In *Computer Vision, 1995. Proceedings., Fifth International Conference on*, pages 666 –671, jun 1995.

[126] B. Kulis. Tutorial on Metric Learning. In *International Conference on Machine Learning*, 2010.

[127] N. D. Lawrence and J. Quinonero-Candela. Local distance preservation in the gp-lvm through back constraints. In *ICML06*, pages 513–520, 2006.

[128] Neil Lawrence. Probabilistic non-linear principal component analysis with Gaussian process latent variable models. *The Journal of Machine Learning Research*, 2005.

[129] J. Lee and T. L. Kunii. Model-based analysis of hand posture. *Computer Graphics and Applications, IEEE*, 15(5):77–86, August 2002. ISSN 0272-1716.

[130] Y. Li and N. S. Pollard. A shape matching algorithm for synthesizing humanlike enveloping grasps. In *Humanoid Robots, 2005 5th IEEE-RAS International Conference on*, pages 442 –449, dec. 2005.

[131] M. Liu, O. Tuzel, A. Veeraraghavan, R. Chellappa, A. Agrawal, and H. Okuda. Pose estimation in heavy clutter using a multi-flash camera. In *ICRA*, pages 2028–2035. IEEE, 2010.

[132] D. G. Lowe. Object recognition from local scale-invariant features. In *ICCV*, pages 1150–1157, 1999.

[133] D.G. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 60(2): 91–110, 2004.

[134] S. Lu, D. Metaxas, D.Samaras, and J. Oliensis. Using multiple cues for hand tracking and model refinement. In *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on*, volume 2, pages II – 443–50 vol.2, june 2003.

[135] M. Malhotra and Y. Nakamura. The relationship between actuator reduction and controllability for a robotic hand. In *IEEE International conference on Biomedical Robotics and Biomechatronics*, 2010.

[136] C. R. Mason, J. E. Gomez, and T. J. Ebner. Hand synergies during reach-to-grasp. *J Neurophysiol*, 86(6):2896–2910, December 2001.

[137] M. T. Mason, S. Srinivasa, A. S. Vazquez, and A. Rodriguez. Generality and simple hands. Technical Report CMU-RI-TR-10-40, Robotics Institute, Pittsburgh, PA, November 2010.

[138] D. Matsui, T. Minato, K. F. MacDorman, and H. Ishiguro. Generating Natural Motion in an Android by Mapping Human Motion. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3301–3308, Edmonton, Alberta, Canada, August 2005.

[139] Microsoft. Kinect. `http://www.xbox.com/en-US/kinect`, 2011. [Online; accessed 31-August-2011].

[140] T. B. Moeslund, A. Hilton, and V. Krüger. A survey of advances in computer vision-based human motion capture and analysis. *CVIU*, 104(2–3):90–126, 2006.

[141] A. Morales, P. J. Sanz, A. P. Del Pobil, and A. H. Fagg. Vision-based three-finger grasp synthesis constrained by hand geometry. *Robotics and Autonomous Systems*, 54(6), 2006.

[142] V. Morariu, B. Srinivasan, V. Raykar, R. Duraiswami, and L. Davis. Automatic online tuning for fast gaussian summation. In *NIPS*, pages 1113–1120, 2008.

[143] G. Mori, S. Belongie, and J. Malik. Efficient shape matching using shape contexts. *PAMI*, 27(11):1832–1837, 2005.

[144] M. Muja. FLANN, fast library for approximate nearest neighbors, http://mloss.org/software/view/143/, 2009.

[145] Richard M. Murray, Shankar S. Sastry, and Li Zexiang. *A Mathematical Introduction to Robotic Manipulation*. CRC Press, Inc., Boca Raton, FL, USA, 1994. ISBN 0849379814.

[146] C. Myers and L. Rabiner. A comparative study of several dynamic time-warping algorithms for connected-word recognition. *Bell System Technical Journal*, 60(7): 1389–1409, September 1981.

[147] C. L. Nehaniv and K. Dautenhahn. Like me?- measures of correspondence and imitation. *Cybernetics and Systems*, 32(1):11–51, 2001.

[148] C. L. Nehaniv and K. Dautenhahn. *The correspondence problem*, pages 41–61. MIT Press, Cambridge, MA, USA, 2002. ISBN 0-262-04203-7.

[149] K. Nguyen and V. Perdereau. Arm-hand movement: Imitation of human natural gestures with tenodesis effect. In *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, pages 1459 –1464, sept. 2011.

[150] V.-D. Nguyen. Constructing force-closure grasps. In *Robotics and Automation. Proceedings. 1986 IEEE International Conference on*, volume 3, pages 1368 – 1373, apr 1986.

[151] C. Nolker and H. Ritter. Visual recognition of continuous hand postures. *Neural Networks, IEEE Transactions on*, 13(4):983 – 994, jul 2002.

[152] OGRE. `http://www.ogre3d.org/`, Last Visited Dec'10.

[153] N. Oliver, A. P. Pentland, and F. Berard. LAFTER: a real-time face and lips tracker with facial expression recognition. *Pattern Recognition*, 33(8):1369–1382, August 2000.

[154] OpenSceneGraph. `http://www.openscenegraph.org/projects/osg`, Last Visited Dec'10.

[155] Otto Bock. Otto Bock at the trade fair 2010 Leipzig. `http://leipzig.ottobock.de/index.php?id=161&no_cache=1&L=1`, 2011. [Online; accessed 31-August-2011].

[156] C. Papazov and D. Burschka. Deformable 3D shape registration based on local similarity transforms. *Comput. Graph. Forum*, 30(5):1493–1502, 2011.

[157] K. Pawelzik, J. Kohlmorgen, and K. R. Muller. Annealed competition of experts for a segmentation and classification of switching dynamics. *Neural Computation*, 8(2):340–356, 1996.

[158] I. M. Pepperberg. *Allospecific referential speech acquisition in Grey parrots (Psittacus erithacus): Evidence for multiple levels of avial vocal imitation*, pages 128–151. MIT Press, Cambridge, MA, USA, 2002. ISBN 0-262-04203-7.

[159] J. Peters and S. Schaal. Reinforcement learning of motor skills with policy gradients. *Neural Networks*, 21(4):682–697, 2008.

[160] PIXAR. Cars. [Movie], 2006.

[161] N. S. Pollard and J. K. Hodgins. Generalizing demonstrated manipulation tasks, November 2002.

[162] N. S. Pollard and V. B. Zordan. Physically based grasping control from example. In Demetri Terzopoulos and Victor Zordan, editors, *ACM SIGGRAPH /Eurographics Symposium on Computer Animation*, pages 311–318, Los Angeles, California, 2005. Eurographics Association. ISBN 1-59593-198-8.

[163] J. Ponce and B. Faverjon. On computing three-finger force-closure grasps of polygonal objects. *IEEE Transactions on Robotics and Automation*, 11(6):868 –881, dec 1995.

[164] P. K. Pook and D. H. Ballard. Recognizing teleoperated manipulations. In *ICRA*, pages 578–585, 1993.

[165] M. Popovic, D. Kraft, L Bodenhagen, E. Baseski, N. Pugeault, D. Kragic, T. Asfour, and N. Krüger. A strategy for grasping unknown objects based on co-planarity and colour information. *Robotics and Autonomous Systems*, 58(5):551–565, 2010.

[166] D. Pratichizzo, M. Malvezzi, and A. Bicchi. On motion and force controllability of grasping hands with postural synergies. In *Proceedings of Robotics: Science and Systems*, Zaragoza, Spain, June 2010.

[167] K. H. Pribham, A. Sharafat, and G. J. Beekman. Frequency encoding in motor systems. In H. T. A. Whiting, editor, *Human Motor Actions. Bernstein Reassessed.*, chapter IIIa, pages 121–156. North-Holland, 1984.

[168] Y. Raja, S. J. McKenna, and S. Gong. Tracking and segmenting people in varying lighting conditions using colour. In *FG*, pages 228–233, 1998.

[169] C. E. Rasmussen and C. Williams. *Gaussian Processes for Machine Learning*. MIT Press, 2006.

[170] C.E. Rasmussen. Gaussian processes in machine learning. *Advanced Lectures On Machine Learning: ML Summer Schools 2003, Canberra, Australia, 2003, Tübingen, Germany, 2003: Revised Lectures*, 2004.

[171] J. M. Rehg and T. Kanade. Digiteyes: Vision-based human hand tracking. In *CMU School of Computer Science*, 1993.

[172] H. Rijpkema and M. Girard. Computer animation of knowledge-based human grasping. *SIGGRAPH Comput. Graph.*, 25(4):339–348, 1991. ISSN 0097-8930.

[173] R. Rosales, V. Athitsos, L. Sigal, and S. Sclaroff. 3D hand pose reconstruction using specialized mappings. In *ICCV*, pages I: 378–385, 2001.

[174] S. T. Roweis and L. K. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 2000.

[175] M. Santello, M. Flanders, and J. Soechting. Postural hand synergies for tool use. In *The Journal of Neuroscience*, 1998.

[176] D. Saxe and R. Foulds. Toward robust skin identification in video images. In *FG*, pages 379–384, 1996.

[177] A. Saxena, J. Driemeyer, and A. Y. Ng. Robotic grasping of novel objects using vision. *International Journal of Robotics Research*, 27(2):157–173, 2008.

[178] Stefan Schaal, AJ Ijspeert, and A Billard. Computational approaches to motor learning by imitation. *Philosophical Transactions of the Royal Society of London. Series B: Biological Sciences*, 2003.

[179] H. M. Schmidt and U. Lanz. *Surgical Anatomy of the Hand*. Thieme, 2003.

[180] S. Schulz, C. Pylatiuk, A. Kargov, R. Oberle, and G. Bretthauer. Progress in the development of anthropomorphic fluidic hands for a humanoid robot. In *Humanoid Robots, 2004 4th IEEE/RAS International Conference on*, volume 2, pages 566–575, nov. 2004.

[181] K. Schwerdt and J. L. Crowley. Robust face tracking using color. In *FG*, pages 90–95, 2000.

[182] G. Shakhnarovich, P. Viola, and T. Darrell. Fast pose estimation with parameter-sensitive hashing. In *ICCV*, 2003.

[183] N. Shimada, Y. Shirai, Y. Kuno, and J. Miura. Hand gesture estimation and model refinement using monocular camera: Ambiguity limitation by inequality constraints. In *FG*, pages 268–273, 1998.

[184] J. Shotton, A. W. 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 *CVPR*, pages 1297–1304. IEEE, 2011.

[185] L. Sigal, S. Sclaroff, and V. Athitsos. Skin color-based video segmentation under time-varying illumination. *IEEE Trans. Pattern Anal. Mach. Intell*, 26(7):862–877, 2004.

[186] D. Song, C. H. Ek, K. Huebner, and D. Kragic. Multivariate discretization for bayesian network structure learning in robot grasping. In *ICRA*, pages 1944–1950. IEEE, 2011.

[187] M. Soriano, B. Martinkauppi, S. Huovinen, and M. Laaksonen. Skin detection in video under changing illumination conditions. In *ICPR*, pages Vol I: 839–842, 2000.

[188] J. Speth, A. Morales, and P.J. Sanz. Vision-based grasp planning of 3d objects by extending 2d contour based algorithms. In *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*, pages 2240 –2245, sept. 2008.

[189] M. Stark, P. Lies, M. Zillich, J. Wyatt, and B. Schiele. Functional object class detection based on learned affordance cues. In *Computer Vision Systems*, 2008.

[190] B. Stenger, A. Thayananthan, P. H. S. Torr, and R. Cipolla. Model-based hand tracking using a hierarchical bayesian filter. *PAMI*, 28(9):1372–1384, 2006.

[191] K. Strobl, W. Sepp, S. Fuchs, C. Paredes, , and K. Arbter. Camera calibration toolbox for matlab. `http://www.vision.caltech.edu/bouguetj/calib\_doc/index.html`, 2011. [Online; accessed 31-August-2011].

[192] E. Sudderth, M. I. Mandel, W. T. Freeman, and A. S. Willsky. Visual hand tracking using non-parametric belief propagation. In *IEEE Workshop on Generative Model Based Vision*, 2004.

[193] E. B. Sudderth, M. I. Mandel, W. T. Freeman, and A. S. Willsky. Distributed occlusion reasoning for tracking with nonparametric belief propogation. In *NIPS04*, pages xx–yy, 2004.

[194] J. D. Sweeney and R. A. Grupen. A model of shared grasp affordances from demonstration. In *Humanoids*, 2007.

[195] CyberGlove Systems. Cyberglove. `http://www.cyberglovesystems.com/all-products`, 2011. [Online; accessed 31-August-2011].

[196] Vicon Motion Systems. Vicon. `http://www.vicon.com`, 2011. [Online; accessed 31-August-2011].

[197] M. Teichmann and B. Mishra. Reactive algorithms for grasping using a modified parallel jaw gripper. In *ICRA*, pages 1931–1936, 1994.

[198] J. B. Tenenbaum, V. de Silva, and J. C. Langford. A Global Geometric Framework for Nonlinear Dimensionality Reduction. *Science*, 2000.

[199] J. C. Terrillon and S. Akamatsu. Comparative performance of different chrominance spaces for color segmentation and detection of human faces in complex scene images, March 15 2000.

[200] A. Thayananthan, R. Navaratnam, B. Stenger, P.H.S. Torr, and R. Cipolla. Multivariate relevance vector machines for tracking. *Lecture Notes in Computer Science*, 3953:124, 2006.

[201] A. Thayananthan, R. Navaratnam, B. Stenger, P.H.S. Torr, and R. Cipolla. Pose estimation and tracking using multivariate regression. *Pattern Recognition Letters*, 2008.

[202] K. A. Thoroughman and R. Shadmehr. Learning of action through adaptive combination of motor primitives. *Nature*, 407:742–747, 2000.

[203] L. H. Ting. Dimensional reduction in sensorimotor systems: a framework for understanding muscle coordination of posture. *Progress in Brain Research*, 165:299–321, 2007.

[204] M.E. Tipping. Sparse bayesian learning and the relevance vector machine. *JMLR*, 1:211–244, 2001.

[205] C. Tomasi, S. Petrov, and A. Sastry. 3D tracking = classification + interpolation. In *ICCV*, pages 1441–1448, 2003.

[206] M. C. Tresch, V. C. K. Cheung, and A. d'Avella. Matrix factorization algorithms for the identification of muscle synergies: Evaluation on simulated and experimental data sets. *Journal of Neurophysiology*, 95(4):2199–2212, 2006.

[207] A. Tsoli and O. C. Jenkins. Neighborhood denoising for learning high-dimensional grasping manifolds. In *IROS*, pages 3680–3685, 2008.

[208] A. Ude, C. Atkeson, and M. Riley. Programming Full-Body Movements for Humanoid Robots by Observation. *Robotics and Autonomous Systems*, 47(2-3):93–108, 2004.

[209] R. Urtasun, D.J. Fleet, and P. Fua. 3d people tracking with gaussian process dynamical models. In *CVPR*, 2006.

[210] R. Urtasun, D.J. Fleet, A. Hertzmann, and P. Fua. Priors for people tracking from small training sets. *IEEE International Conference on Computer Vision (ICCV)*, pages 403–410, 2005.

[211] N. Vahrenkamp, S. Wieland, P. Azad, D. Gonzalez, T. Asfour, and R. Dillmann. Visual servoing for humanoid grasping and manipulation tasks. In *8th IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, pages 406 –412, dec. 2008.

[212] F. J. Valero-Cuevas, F. E. Zajac, and C. G. Burgar. Large index-fingertip forces are produced by subject-independent patterns of muscle excitation. 31:693–703+, 1998.

[213] J. M. Wang, D. J. Fleet, and A. Hertzmann. Gaussian process dynamical models for human motion. *PAMI*, 2008.

[214] R. Y. Wang and J. Popovic. Real-time hand-tracking with a color glove. *ACM Trans. Graph*, 28(3), 2009.

[215] A. S. Weigend, M. Mangeas, and A. N. Srivasta. Nonlinear gated experts for time series: Discovering regimes and avoiding overfitting. *Int. J. Neural Syst.*, 6:373–399, 1995.

[216] Y. Wu and T. S. Huang. View-independent recognition of hand postures. In *Computer Vision and Pattern Recognition, 2000. Proceedings. IEEE Conference on*, volume 2, pages 88 –94 vol.2, 2000.

[217] Y. Wu, J. Y. Lin, and T. S. Huang. Capturing natural hand articulation. In *ICCV*, pages 426–432, 2001.

[218] J. Yang, W. Lu, and A. Waibel. Skin-color modeling and adaptation. *Lecture Notes in Computer Science*, 1352:687–??, 1997.

[219] M. Yang and N. Ahuja. Gaussian mixture model for human skin color and its applications in image and video databases, November 09 1999.

[220] B. Yao and L. Fei-Fei. Modeling mutual context of object and human pose in human-object interaction activities. In *CVPR*, 2010.

[221] X. Zhao, H. Ning, Y. Liu, and T. Huang. Discriminative estimation of 3d human pose using gaussian processes. In *ICPR*, 2008.

[222] H. Zhou and T. S. Huang. Tracking articulated hand motion with eigen dynamics analysis. In *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*, pages 1102 –1109 vol.2, oct. 2003.

[223] H. N. Zhou, D. J. Lin, and T. S. Huang. Static hand gesture recognition based on local orientation histogram feature distribution model. In *Vision for Human-Computer Interaction*, page 161, 2004.

[224] X. J. Zhu, J. Yang, and A. Waibel. Segmenting hands of arbitrary color. In *FG*, pages 446–453, 2000.