HANDLE

Developmental pathway towards autonomy and dexterity in robot in-hand manipulation

Deliverable 31

Developmental methods in exploratory learning

Due date of deliverable: **Month 48**
Actual submission date: **Month 48**
Partner responsible: UHAM

Version 1.0

Classification: PU
Grant Agreement Number: 231640
Contract Start Date: 2009-02-02
Duration: 48 Months
Project Coordinator: UPMC
Partners: UPMC, SHADOW, UC3M, FCTUC, KCL, ORU, UHH, CEA, IST
Project website address: www.handle-project.eu

Revision History

Date	Version	Change	Author
2012.06.08	0.1	created	Hendrich
2013.01.18	0.2	action-gist directed babbling	Cheng
2013.01.19	0.3	babbling for grasping	Bernardino
2013.01.24	0.6	updates	Zhang
2013.01.25	0.7	submitted for partner review	Hendrich
2013.01.31	1.0	revision and final version	Hendrich

Contents

List of Figures

List of Tables

List of Algorithms

1 Overview

This report summarizes the current research and algorithms about *developmental methods in exploratory learning* in the HANDLE project. This is the final report of work-package WP3 *improving skills*, where exploratory robot actions are used to learn about objects and their affordances, in order to enhance the robot's performance for both grasping and manipulation. The idea is to generalize the robot's current grasping and handling knowledge to novel situations including unknown objects, and to improve existing robot knowledge by self-exploration and measuring the effects of the actions.

However, exploration learning in high dimensional spaces is typically intractable unless "smart" exploration strategies are applied; one instance of Bellmann's famous *curse of dimensionality*. Therefore, the project has researched and developed methods to reduce the complexity of exploration learning in manipulation tasks. As noted in the project work-plan, the development of grasping in young human infants provides one promising starting point for research into suitable algorithms. Unfortunately, the extensive motion babbling and learning used by infants during years of growing is simply not possible given the current state of hardware development, where multi-fingered hands are still rather fragile. Another problem concerns the sensing, as the correlation of robot actions to their effects on the manipulated objects requires detecting and tracking those objects with sufficient accuracy. The recent project reports D18 *Visual and tactile perception algorithms for grasping* [146] and D23 *Visual and tactile perception system evaluation report* [151] have summarized the algorithms studied or developed to track the object and hand state by fusion of visual and tactile information; but realiable and robust tracking of small objects during in-hand manipulation tasks is still an unsolved problem.

Therefore, another line of work is based on simulation environments like Graspit! [24], Gazebo/ROS [63], or Openrave [74], where all properties of the simulated robot hand as well as the grasped objects and the enviroment are easily accessable. The approach finally adopted by the project is largely based on the use of simulation for the initial development of algorithms and testing on a large number of objects, with selected key experiments performed on the real robots and the Paris demonstrator platform to prove the implementation of the algorithms. Of course, the use of grasp simulators and physics engines is typically restricted to the point-contact model, while more realistic soft-finger models are not supported by the current generation of simulators. Also, accurate 3D-models of the hand and all studied grasp objects are required, which implies an adaptation step for application on the real system, where calibration errors and object reconstruction from noisy sensors limit the accuracy.

The algorithms for exploration learning described in this report target the use of the Shadow hand for concrete manipulation motions given a particular task. The use of babbling in discrete spaces and clustering for discovery of object affordances is

the topic of a companion report D30 *Discovering new affordances from behavioral babbling* [154] which in turns builds upon the modeling and extraction of affordances as reported in D14 *Improving known actions from motor babbling* [142]. In the latter report, the concept of *action space metrics* was introduced, to measure the effect of robot actions directly by calculating similarity measures in motor-terms, e.g. measured joint-angles defining the hand-pose or tactile-sensor readings.

1.1 Outline of this report

The remainder of this report is structured as follows.

- First, section 2 summarizes the *software infrastructure and experiment setup*, including a description of the equipment used. This includes complete Gazebo simulation models for the Shadow hand and arm systems in Paris and Hamburg. Several new ROS nodes were added to the existing ROS system, in order to use the Handle manipulation stack for the experiments. In addition to Gazebo, OpenRave was also used for the simulations reported below in section 4, as it provides simulation models for the Barrett- and Shadow-hands.

  Regarding object tracking on the real demonstrator platforms, the Kinect is used together with standard cameras and stereo-cameras. Unfortunately, tracking robustness and accuracy for in-hand manipulation proved to be insufficient for several tasks, especially with the fingers caging small objects. Marker-based techniques would remedy this situation to some extent, but the approach described in section 2.4 falls back on the original project idea of complementing the built-in sensing of the Shadow hand with *instrumented objects* carrying their own sensors and providing additional sensing modalities beyond proprioception (joint-angles) and tactile sensing.

- Next, section 3 presents the current development of *postural synergies for grasp-planning* and the execution of in-hand manipulation motions with the Shadow hand. As motivated in report D24 *Parameterizing and creating new actions* [152], synergies are one of the most promising techniques to reduce the enormous search-space of grasp-planning with multi-fingered hands. For the Shadow hand, the full configuration space has 24-DOF of which 20-DOF are controllable, with another 6-DOF required to specify the hand-object pose. On the other hand, the initial analysis presented in [152] indicated that only 1..6-DOF (parameters in eigenspace) were required to reconstruct the grasp poses from the human demonstrations with sufficient accuracy. In this report, we study the use of the postural synergies for grasp-planning and in-hand manipulation motions. To this end, section 3.1 first summarizes the relevant concepts about postural synergies, and also describes the results of an additional learning session, where kinesthetic teaching was used to perform

human-like grasps on the UPMC demonstrator platform. This experiment set complements (and confirms) the results previously obtained using cyber-glove tele-operation recorded on the UHAM air-muscle hand. In the next subsection 3.2.1, we present an analysis to reconstruct the grasp origin (grasp center point) for each of the eight grasp-classes recorded, and also correlate the grasp origin with the size (and shape) of the objects used for training. This gives us both the hand pre-shape and the final hand-pose for grasping an object of given size, plus the approach direction and the hand+arm target point for reach-to-grasp motions. To compensate for errors in the estimated object size and pose, we propose an adaptive *reactive* grasping strategy, which is explained in subsection 3.3. The following subsections present selected example manipulation motions based on different grasp-types in detail.

- The next section 4 is concerned with *motion babbling to improve the quality of grasps* via active exploration of the target object. First, subsection 4.1 summarizes the classic contact models and desribes the Bimodal Wrench Space analysis selected for the evaluation of grasp quality. The concept of Bayesian optimization and Gaussian Process regression is then applied to the grasp optimization problem in subsection 4.2. We explain that the algorithm is still feasible despite the high-dimensionalty of the search space implied by the Shadow hand, given that uniform sampling of the search space is replaced by a smart exploration technique. Experimental results from simulation and the real hand are presented in section 4.3 for the Barrett-hand and the Shadow-hand and a set of prototypical grasp objects. Initial results demonstrate that the algorithm can also be applied to improve grasping based on the postural synergies for the Shadow hand.

- The last section 5 is based on the *action-gist concept* for generation of *in-hand manipulation motions*, where human demonstrations are analyzed to identify and classify the key motions, see [142] for an overview and [94] for subsequent refinements. In order to generate finger motions for the Shadow hand for in-hand manipulation, the algorithm uses Particle Swarm Optimization learning to search for and adjust a solution. First results from Gazebo simulation are presented for object in-hand rotation, including finger gaiting.

- The report concludes with a short summary and the list of references.

2 Experiment Setup

This section describes the software architecture and hardware setup used for the simulations and experiments. First, an overview of the software architecture used for learning and execution is presented in subsection 2.1. Almost all software used by the project is now integrated within the ROS framework, allowing us to run the same software in simulation and when using the real hand. New interface nodes were developed to also bridge the older air-muscle hand available in Hamburg into the ROS framework. The sensing setup is described in subsection 2.3 and the use of instrumented objects for improved pose-tracking is explained in subsection 2.4. Finally, subsection 2.5 describes a few key details of the simulation setup in ROS/Gazebo.

2.1 ROS software overview

The overall software architecture used for the simulations and experiments described in the next chapters of this report is sketched in figure 1. All key software modules are now available as ROS nodes and services, building on the huge base of public ROS stacks and packages. This ranges from the basic communication and build-infrastructure provided by ROS itself via standard libraries like OpenCV and PCL to the large number of specific components developed by the HANDLE project partners [156]. For example, the Orocos library provides the data-structures and tools for generic forward and inverse kinematics calculations, while a module developed at UPMC uses the library for FK/IK services which respect to the J1/J2 joint-coupling on the C5/C6-type Shadow hands.

The updated Shadow stack includes a set of utilities and tools for modeling and controlling the Shadow hands, while the Shadow EtherCAT stack provides the fast and high-bandwidth interface to the C6 motor hands. However, there is still a need to interface to legacy software, and a set of ROS nodes was developed to integrate previously developed software into the ROS HANDLE manipulation stack:

- The Xacro/URDF descriptions of the robots are a key component for many ROS functions. The ROS/URDF models for the Shadow hand were updated for compatibility with ROS Fuerte and newer releases of the Gazebo simulator. A URDF model was developed for the new Shadow hand with Syntouch BioTAC tactile sensors.

- The *MuscleHandProxy* node provides the same interface (published and subscribed topics) as the Shadow EtherCAT drivers for the C6 motor hand and the Shadow hand Gazebo interface. It connects via TCP/IP to its companion server *CANbusServer*, which in turn interfaces to the Linux/RTAI/CAN-bus software used to control the older generation of CAN-bus based Shadow

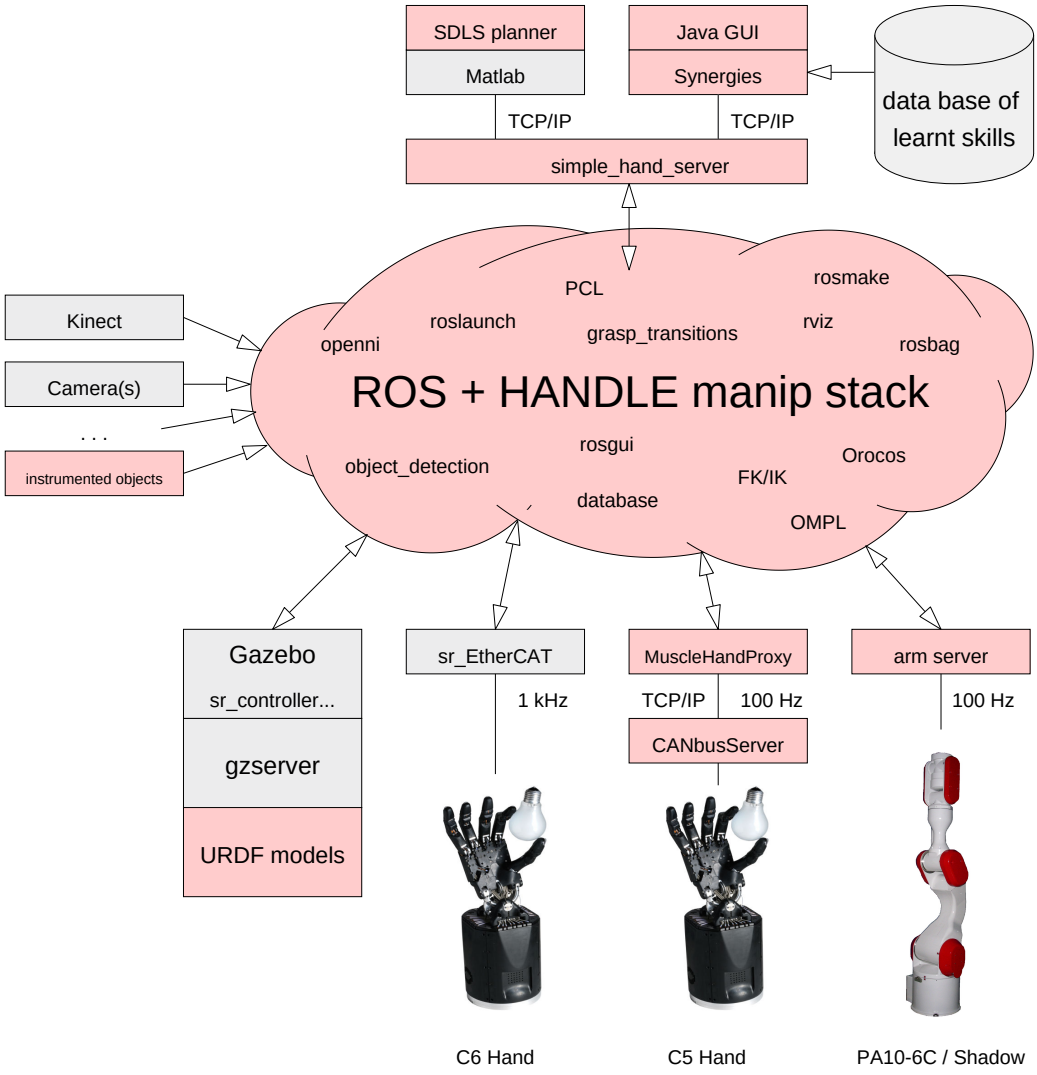

Figure 1: Overview of the software architecture used in the experiments, now completely centered on ROS and the HANDLE manipulation stack. A set of interface nodes have been written to integrate legacy hardware and tools into the ROS universe. Software modules developed or updated for the report are marked in red.

hands. However, the CAN-bus system has a much lower bandwidth than the current generation EtherCAT systems, which limits the achievable publish-rate for joint-angle and tactile-sensor data. Also, *CANbusServer* includes fine-tuned algorithms to limit the rate of setpoint commands sent to the CAN-bus hand.

- The *SimpleHandServer* node provides the interface between our legacy software and the ROS manipulation stack. However, only a fixed set of ROS topics and services is supported, subscribing to the ROS joint-position and tactile-data from the hand(s), and publishing to the joint-position controllers. This node allows us to run the existing Java GUI for grasp-recording, interpolation, and synergy-based control on top of the ROS framework. It also provides a means to access recorded grasps based on the HANDLE database format [134] in addition to rosbag files recorded within ROS. SimpleHandServer has also been used by us to connect Matlab to ROS for hand control, including experiments with the interative SDLS full-hand inverse-kinematics solver developed at UC3M.

2.2 Demonstrator platforms

The experiments described in this report have been performed on two different demonstrator platforms as well as in simulation. The main platform is the project demonstrator setup at UPMC in Paris, with the Shadow C6 EtherCAT hand mounted on the Shadow 4-DOF air-muscle arm. A detailed description of the system was published in [138]. Additional experiments were performed using the setup in Hamburg, with the Shadow C5 air-muscle hand mounted on a Mitsubishi PA10-6C robot arm, see [152] for a description. For both platforms, complete ROS URDF/Xacro descriptions including the hand with tactile sensors, the robot arm, and the Kinect cameras are available.

2.3 Visual and tactile sensing

The sensing setup is very similar for both platforms, with a Microsoft Kinect used as the main sensor. However, while the Kinect is mounted on top of the arm in Paris, a fixed wall-mount position has to be used for the Kinect at Hamburg. This requires corresponding changes in the ROS URDF descriptions of the robot models, and some changes in the setup of the table- and object-detector ROS nodes. Additional cameras can be included in the sensor setup.

While the standard ROS nodes for tabletop and object cluster detection are quite robust and accurate, no satisfactory solution was found for the pose-tracking of objects grasped by the Shadow hand. Several approaches including textured objects together with SIFT, and QR-marker based tracking were tested, but the occlusion

problem was not solved. As shown in figure 2 (left), color-based markers were also tried in Hamburg to improve object-pose tracking. In order to connect the three-markers-cross easily to multiple objects, a Lego brick is used. However, changing illumination conditions and the required complex color-calibration made the overall system impractical.

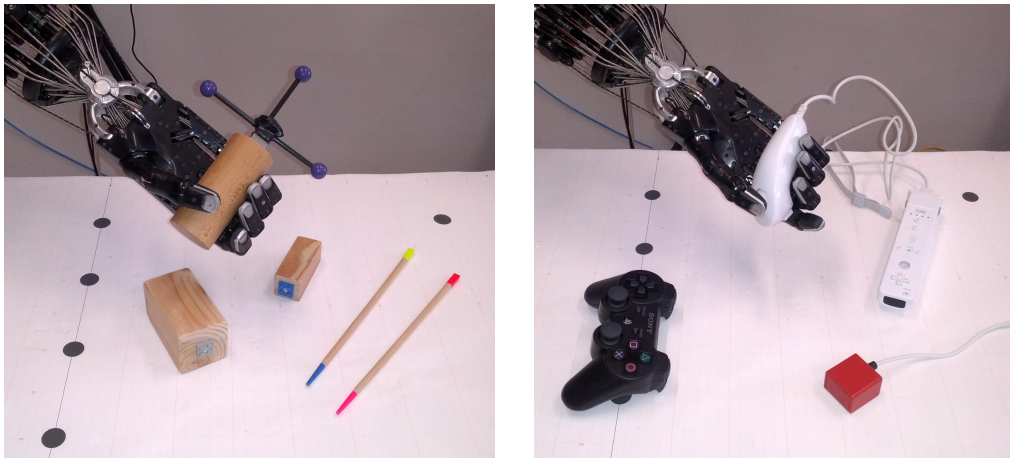

Figure 2: Marker-based and instrumented objects used for improved object tracking for in-hand manipulation. Left: objects with markers for visual tracking. Right: The Nintendo Wiimote controller, Nunchuk controller, Sony Sixaxis, and the Codemercs JoyWarrior devices all integrate 3-axis accelerometers. This allows the direct reconstruction of the roll and pitch pose of the object, even when visual information is not available or not reliable.

2.4 Instrumented objects

To associate the effects of robot actions with the corresponding movements of the grasped and manipulated object, the pose of the object needs to be tracked. However, as explained above, this is proved to be difficult with the available external sensors including stereo-cameras and the Kinect. Therefore, we returned to the project idea of using *instrumented sensing objects* to complement the robot sensor with data provided and recorded by the manipulated objects themselves. Of course, a commercial object tracker like the Polhemus system [56] would be an obvious solution, but the system is not available on the demonstrator platforms. Also, a magnetic tracker like the Polhemus has difficulties when operating near to large metal objects like the PA10-6C robot arm.

Due to the high cost and effort to design custom instrumented objects, we evaluated a set of readily available consumer electronics devices with integrated sensors. The first group of instrumented objects is shown in figure 2 (right), where all four devices include a built-in 3-axis accelerometer. The white object on the table is

the Nintendo Wiimote controller, while the Shadow hand grasps the companion Nunchuk joystick. On the left of the table is the Sony Sixaxis joystick, and the small orange object on the right is a Codemercs Joywarrior. The three game controllers also provide a set of buttons and analog (continuous) switches or joysticks, which can be used to learn tool-use and in-hand manipulation motions.

Assuming that the object is at rest, a simple calculation of the accelerations along the (x, y, z)-directions of the sensors provides immediate information about the object pose, and therefore tracking data required for the evaluation and adjustment of in-hand rotation motions. A falling object is easily detected, because the total measured acceleration is zero in this case. Object slippage can be detected as well in many cases, but this has not been exploited in the experiments described below. Of course, the low-cost miniature accelerometers used in all four devices are by far not precise enough to calculate the full object pose from the accelerations and a given initial position.

2.4.1 Nintendo Wiimote controller

The Wiimote (Wii remote) controller is the main human input device for the Nintendo Wii gaming console. The base system features a set of push-buttons, but no analog or continous switches or joysticks. Instead, the built-in 3-axis accelerometer is used as the main sensor for gaming control. A custom on-board infrared camera tracks up to four external light-sources, which can be used to provide ground-truth for the object pose. A connector on the rear end of the controller allows us to plug in a set of extensions, notably the Nunchuk companion controller and the motionplus device which includes a 3-axis gyroscope. The device connects to the host via Bluetooth, and its basic communication protocol has been reverse-engineered. The *wiimote* ROS package provides access to the raw-data sent by the controller.

The Wiimote controller can be grasped by the Shadow hand easily, but not all buttons of the controller can be reached by the thumb and index finger. Depending on the grasp pose chosen, at least the 4-way directional pad or the main trigger buttons can be reached and pressend by the Shadow hand.

The Nunchuk joystick is one of the extensions of the main Wiimote controller, and is connected to the main controller via a cable. It provides a 2-axis joystick, two trigger buttons, and the built-in 3-axis accelerometer. As shown in figure 2 (right), the Shadow hand can grasp the Nunchuk quite easily, with the trigger buttons in reach by the index fingers, and the joystick just in reach of the thumb.

2.4.2 Sony Sixaxis joystick

The Sony Sixaxis joystick is the main input device for the Playstation 3 gaming console. The controller includes two analog 2D-joysticks, two analog switches/triggers, a set of push-buttons, and the 3-axis accelerometer. Connection to the host is

possible either via USB cable or via Bluetooth. The communication protocol with the host has been reverse-engineered and drivers for the device are included in recent Linux kernels. As the Sixaxis is used as the main remote-controller for the WillowGarage PR2 service robot, a ROS driver which supports all functions of the controller is available in the *ps3joy* package.

Unfortunately, the controller is designed for two-hand operation, and humans hold the controller using the flesh on the base of the thumb, which is not available on the Shadow hand. As it turned out, the controller simply cannot be grasped by the Shadow hand in the position used by humans, which limits the use of the Sixaxis as an instrumented object to unrealistic grasps on the center of the controller.

2.4.3 Codemercs JoyWarrior

The JoyWarrior24F device (www.codemercs.com) integrates a 3-axis accelerometer with user-selectable sensitivity, providing 14-bit values at a sample-rate of up to 125 Hz. Host connectivity is via a standard USB cable, where the device identifies itself as a 3-axis joystick, so that no external drivers are required. However, a quick calibration of the device is recommended. On Linux systems, it will be necessary to override the default kernel settings which apply a deadband to the presumed joystick null position, canceling out data when the accelerometer is oriented near horizontally. Integration into ROS using the joy_node stack is straightforward.

With a weight of about 5 gr and a size of just 30x33x5 mm^3, the device can be integrated into or attached to many target objects. For the experiments reported below, the optional plastic casing sold by Codemercs was used, which results in a small orange box of size approx. 35x37x25 mm^3.

2.4.4 iControlsPro

The *ICON iControlsPro* MIDI controller is a low-cost human-interface for audio workstation software. It provides a set of nine motorized faders (7-bit resolution), nine incremental knobs, one master jog-wheel, and a large set of buttons. The host connection is via USB, where the device is automatically detected and recognized as a MIDI controller without the need of additional custom device drivers. The mapping from the hardware sensors (buttons, knobs, faders) to the sent MIDI messages is user-configurable, and several presets are provided as part of the Windows-based driver software. However, we bypass the provided presets and have developed a small Java library instead for better flexibility. Our library connects to the device on either Linux and Windows, and maps each MIDI event to user-specified callback functions.

The device has been used to learn manipulation motions, where the MIDI events that are triggered when either moving the faders or knobs or when pressing and

Figure 3: The iControlsPro MIDI controller is a low-cost input device for audio workstation software. It provides motorized faders plus a set of buttons and knobs, and has been used by us in two functions: First, as an efficient human-interface device for interactive hand control, and second, for tracking the effects of robot motions for exploration learning.

releasing the buttons, provide the learning system with direct feedback about the device state, without the complexity of visual object pose tracking.

In addition, we have used and studied the device as an improved human-computer interface for interactive control of the Shadow hand. For example, using the faders and knobs for direct joint-level control is a nice improvement over the traditional mouse-based user-interfaces using sliders, because it is possible to feel the different faders and to find them by touch, without having to look at the computer monitor. Using the motorized faders, the faders can be set to a given initial position when switching modes, for example when switching between joint-level and synergy-level control.

2.5 Simulation environment

One of the major advantages of the ROS framework is the seamless integration of the Gazebo multi-robot simulator. The underlying ODE physics engine provides a base-line physics simulation that is sufficiently accurate for simple grasping tasks, even if the support for detection and simulation of multiple contacts between two objects is less than perfect. Sensor plugins enable the simulation of a variety of cameras and sensors, including a robust and realistic simulation model of the Kinect sensor.

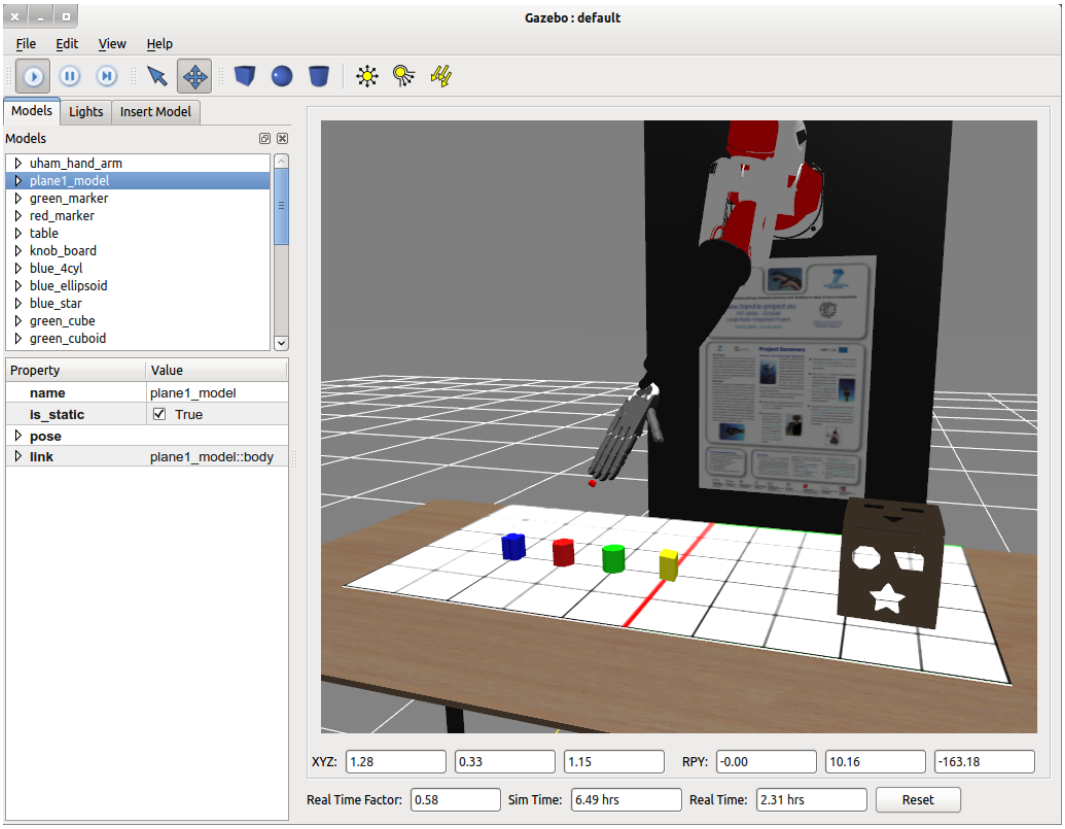

Figure 4: Gazebo simulation setup for the Hamburg setup. Shadow C5 air-muscle hand on wall-mounted PA10-6C robot arm, experiment table with toy-sorting box objects, and Kinect sensor.

The Gazebo simulator can import most existing ROS Xacro/URDF models, including the URDF model of the Shadow hand and Shadow arm provided as parts of the Shadow robot ROS stack. Complete simulation models of the experiment platforms in Paris and Hamburg have been developed, using the arm+hand calibration of the robot with respect to the world coordinate system. A set of grasp objects has been made available as part of the *handle_objects* ROS stack. This includes a simulation model of the iControlsPro MIDI controller with moving sliders and knobs. Also, the ROS models of the Shadow hand were updated for compatibility with the Fuerte release of Gazebo. Figure 4 shows a screenshot of the simulated Hamburg setup, with four objects and the toy-sorting-box on the table.

The Shadow stack uses the same nodes and topic names for controlling either the real EtherCAT hand or the simulation model in Gazebo. Measured hand-state information from the real hand uses the same data-types and message names as the data extracted from the Gazebo simulator using the *sr_controller_manager* plugin. Therefore, higher level functions including grasp learning can be used without any change at all either in simulation or on the real platform.

3 Postural Synergies for Manipulation

In this section, we describe the use of postural finger synergies for grasp-planning and the execution of manipulation motions. The fundamental idea is based on the observation of human grasping, where most hand motions lack individuation in finger movements, and only a small part of the whole hand configuration space is actually used by humans. A successful transfer of this concept to multi-fingered robot hands is a very promising approach towards solving the grasp-planning problem, because only a small subset of the overall state-space of the hand has to be searched for successful grasp poses. Please refer to the project report D13 *Algorithms for planning the grasping of objects for manipulation and for planning the in-hand manipulation* (chapter V) [141] for an up-to-date introduction, including a review of relevant literature from neuroscience and robotics.

A seminal analysis of postural synergies for grasping was described in the classic study [20], where Santello et. al. asked several test subjects to shape their hands as if to manipulate imaginary everyday objects, while the hand poses were recorded with a data-glove. The study demonstrated that the fingers were shaped using certain common patterns, despite of intersubject variations. A Principal Component Analysis of the recorded data showed that the first two principal components accounted for more than 80% of the variance, strongly suggesting that the grasp postures used by the humans could be approximated by a 2-dimensional basis instead of the 22-dimensional basis required to describe all 22-DOF typically assigned to the human hand. This fact is also reflected in the classic grasp taxonomies [5], where only a handful of different poses are sufficient to explain the hand motions used by humans for grasping.

Ever since the Santello study [20], the use of postural synergies for robot grasp planning has attracted a lot of interest, mostly in the context of static grasping of unknown objects. An important milestone was reached when Ciocarlie et. al. integrated their so-called Eigengrasp-planner [22] into the GraspIt! simulation framework [24]. The Eigengrasp planner includes hand-crafted synergy matrices for a set of different robot hands as well as the original matrices from the Santello study adapted to the Graspit! model of a humanoid hand. A configuration dialog lets the user load different synergy matrices and enable or disable any of the Eigengrasps for the grasp planning. When restricted to about 2..3-DOFs for the in-hand pose, the planner has to consider only the 6-DOF corresponding to the relative hand-object pose, and is often fast enough to compute novel grasps on complex objects described by 3D-meshes within tens of seconds. Using human demonstrations of the 6-DOF grasp pose (hand-object distance and approach vector), real-time grasp adjustment has been demonstrated, and is currently proposed as a realistic tool to help injured and handicapped people.

However, the problem of deriving suitable synergy vectors for multi-finger hands and complex kinematics remains unsolved. As mentioned above, so far only ad-hoc

solutions have been used for the robot hands supported in the GraspIt! simulator. The idea is to start with basic close-finger motions, and to also exploit symmetries in the hand-structure. For example, the first synergy defined for the three-finger Barrett hand uses synchronous finger-closing of all three fingers, while the palm-spread motion is used as the second synergy.

3.1 Learning Postural Synergies from Human Demonstration

One promising approach towards the learning of suitable postural synergies for the Shadow hand from human demonstrations was proposed in deliverable D24 *Parameterizing and creating new actions* [152]. Exploiting the dexterity of humans, it is straightforward to record a large number of grasps for a carefully chosen set of test objects using a data-glove. This can be performed quickly and results in a rich database of human demonstrations. However, the problem of mapping those human grasps to the different kinematics of the robot hand remains, and so far remains unsolved. Therefore, we decided to simply bypass the mapping problem by using a tele-operation approach, where human experimenters performed a set of grasps on the carefully chosen set of test objects, but using the actual robot hand. A grasp is only recorded and added to the database when the experimenter accepts the grasp pose as *human-like*, building a data-set of dexterous grasps poses for the robot hand.

This tele-operation approach was carried out in early 2012 to record several complete sets of human-like grasps using the Hamburg air-muscle hand. The experiments concentrated on eight different dexterous fingertip grasp-classes (palmar pinch, tip pinch, tripod, writing tripod, lateral, lateral tripod, addiction grip, parallel extension) and used a set of demonstration objects of basic shapes (sphere, cylinder, boxes) of different sizes. Running a Principal Component Analysis on the recorded grasps then provided us with the synergy matrices, as well as correlations of grasp types with object size and shape, and therefore a first correlation to extract object affordances. Please refer to the project report D24 [152] for details and the initial analysis.

However, there remained a doubt whether the grasp-poses recorded in Hamburg on the Shadow C5 air-muscle hand would also be valid on the newer Shadow C6 motor hand in Paris, because of differences in the hand kinematics. While both hands are based on the same basic kinematics structure, the joint-limits of the hands are slightly different, and the hands use tactile-sensors on the fingertips with different geometry. More importantly, the J1/J2-coupling between the driven medial finger joints and the underactuated distal finger joint is quite different, with an almost linear coupling ($J1 = 1.1 \cdot J2$) on the C5 hand, and a highly non-linear joint-coupling on the C6 hand. Due to both mechanical troubles and the difficulties encountered when trying to model the non-linear behaviour, the tendon-based joint-coupling on the Paris C6 hand has been replaced by a simple mechanical struct in the meantime, resulting in an exact 1:1 J1/J2 coupling.

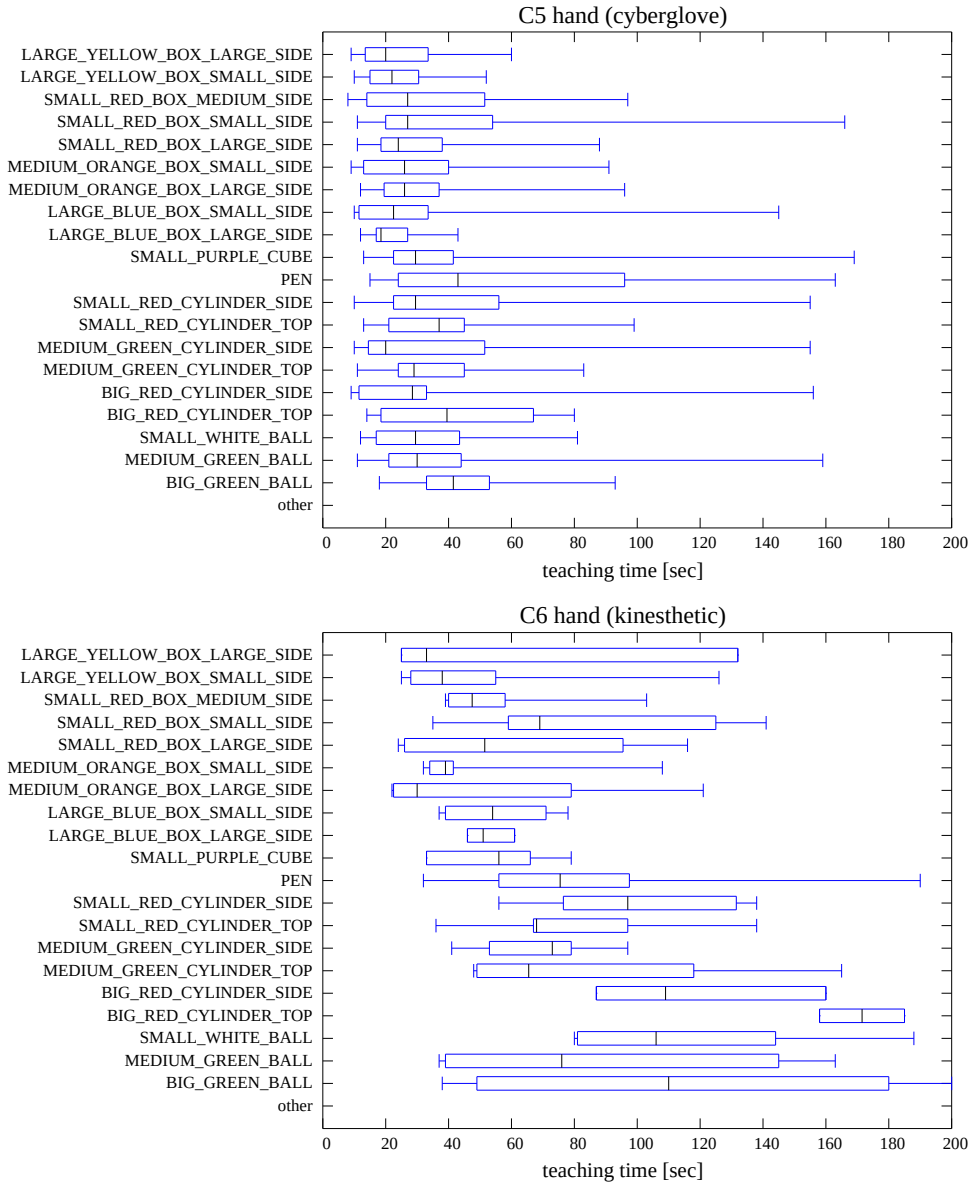

Figure 5: Teaching time used until the experimenters accepted the resulting grasp pose as human-like. Above: cyberglove tele-operation on the UHAM C5 air-muscle hand, Below: kinesthetic teaching on the UPMC C6 motor hand (zero-force mode). Overall, the dataglove tele-operation approach turned out to be much faster than kinesthetic teaching.

Therefore, another human demonstration recording session was performed using the project demonstrator hand in Paris, where human-like grasp poses were recorded for the same set of test objects and the same eight grasp classes. However, this time *kinesthetic teaching* was used to adjust the fingers to human-like grasp positions instead of the cyberglove tele-operation employed during the previous recording sessions in Hamburg. This was tested mainly for two reasons. First, kinesthetic teaching is an interesting option that was not available on the Hamburg air-muscle hand, which only provides joint position controllers but no direct force-control mode. Second, the experimenters were unhappy with the orientation of the thumb for several grasps when using the cyberglove tele-operation in Hamburg, and kinesthetic teaching promised a more direct way to adjust the thumb position. This was done with two experimenters holding the test object and then adjusting the fingers around the object, while a third experimenter switched the hand mode to zero-force mode for grasp adjustment and recorded the grasps. As it turned out, the recorded grasps and therefore also the postural synergies extracted from the human demonstrations are very similar for both hands. The differences between the J1/J2 couplings of the fingers were only relevant for a few grasp-types and certain object sizes. The important kinematics of the thumb is the same on both version of the hand.

However, the situation is quite different regarding the time required by the experimenters to perform the grasps. The boxplots in figure 5 indicate the time taken by the experimenters to grasp the different objects, that is the time until a grasp pose was accepted as human-like and then recorded and added to the dataset. While this information is not directly available from the recorded grasp pose datasets, the time taken for grasping an object can be reconstructed from the timestamps of the grasps, as the grasp class and object type are available in the grasp annotations. Some implausible (long) times cannot be explained by this, and the corresponding data points have been removed from the data shown.

It is obvious from the data that training with the calibrated cyberglove is much faster than kinesthetic teaching, at least for the vast majority of grasps. Of cousre, the cyberglove allows the human experimenter to exploit a live-long experience of finger-motions, but it also shows that the cyberglove calibration did work reasonably well, and many grasp poses were reached and accepted after only a few seconds. For some grasps and objects, however, the fine-tuning of the finger poses and especially the adjustment of the thumb position took a relatively long time before the experimenters were happy with the hand poses.

3.2 Synergies for grasp planning

The recorded datasets contain a large number of grasps for objects of different size and shape. The postural synergies extracted from the datasets provide a nice parametrization of the hand configuration space, and finger poses for a novel object of given size can be calculated efficiently. However, additional information is

required to create a complete grasp-planning system that is able to reach for an object and then grasp it. This includes the calculation of the required grasp origin, that is the relative pose $dP = (\delta x, \delta y, \delta z)$ between the center of the object and the palm-coordinate frame of the hand. Once this information is available, traditional inverse kinematics calculations can be performed to find solutions for the arm and wrist configuration to reach a given object at position (x, y, z) in the world coordinate system. For the reach-to-grasp motion we also have to find a suitable approach direction of the hand towards the object. The reconstruction of the hand-object pose and estimation of approach vectors is described in the next two subsections.

3.2.1 Grasp center point and approach vectors

As mentioned above, no absolute object tracking system (e.g. Polhemus) was available for the human demonstration recordings. This would have allowed us to put markers on the test objects and directly record the required information about the relative pose between objects and the palm.

Therefore, we have to recreate the hand-object pose from the existing datasets. All objects used during the training sessions are simple geometric shapes (sphere, cylinder, box), and the exact size of the objects is known. We now assume that all experimenters did in fact adjust the grasps to be symmetrical according to the object shape, before accepting the grasps as human-like. We also assume that all fingers taking part in a grasp according to the given grasp class did actually contact the object. Under these two conditions, we can first run forward-kinematics on the recorded grasp poses to find the cartesian coordinates of the phalanges and the fingertips of the hand, and then find the geometric center between the fingertips used in the grasp. Of course, for the lateral and lateral-tripod grasps, the side of the first-finger is used as the relevant coordinate frame in the calculations.

The result of this analysis is shown in figure 6, where the reconstructed hand-object pose is plotted for all grasps of the recorded dataset, and markers indicate the eight different grasp types. The (x, y, z) coordinates are indicated by colors (red, green, blue) in turn and plotted against the object-size estimated from the distance between the fingertips of the thumb and first finger.

A right-handed coordinate system aligned with the fingers is assumed:

- the *origin* $(0, 0, 0)$ is at the WRJ1 joint of the Shadow hand
- x runs to the left (along the stretched thumb)
- y is up (from the palm),
- z runs along the palm from the wrist to the tip of the middle finger.

As was to be expected, the reconstructed grasp center points depend on the grasp type as well as the object size. However, the rather small variance for the x coordinates is surprising, given that the thumb of the Shadow hand has a range of

almost 200 mm. The lower subfigure shows an even more close clustering of the x coordinates, where a subset of the whole dataset has been used from only three grasp-classes ($\square$: writing tripod, $\circ$: parallel extension, $\triangleright$ lateral).

Figure 7 on page 20 shows a more detailed analysis for six of the eight recorded grasp-classes, with addiction-grip omitted due to the small dataset and parallel-extension being similar to tripod. Again, the (x, y, z) components are plotted in (red, green, blue) color, while the object size is now taken from the known dimensions of the test objects, and markers correspond to the experimenters. It is obvious that some experimenters preferred slightly different grasp poses (e.g. tip-pinch), but overall the demonstrated hand poses are very similar.

The figure also plots the linear regression through the dataset, averaged over all demonstrations. See table 1 on page 21 for the numerical values. This provides us with the (x, y, z) components of the relative hand-object pose as a function of object diameter for all of the recorded grasp types. Using the approximation, the relative hand-object pose does not need to be hardcoded in the object description (like done in the ROS manipulation stack database), but can be calculated on-the-fly for any given object with known or approximately known size.

The approach vectors for reach-to-grasp motions are more difficult to estimate, as humans are known to use complex motions which consider obstacles and the task context. The current solution is to use fixed approach vectors which are orthogonal to the main close-grasp direction for the eight grasp-types. For example, a tripod grasp touches the target object with the thumb, index finger, and middle finger, and the object orientation will be parallel to the x-z plane. Therefore, y is the suitable approach direction. For the lateral grasps, the human demonstrations show the largest variance, as object of middle size can be grasped in many different orientations. Here, we select an approach vector slightly along the x-z plane, with the hand mostly moving forward but also a bit to the left. Table 2 gives the approach vectors selected for the eight grasp classes studied. Again, the approach vector is considered a property of the grasp type, and as such only needs to be included in the grasp database for objects that impose extra geometric constraints on the reach motion.

3.3 Reactive approach and grasping

Given a grasp-class, hand pre-shape, and the approach vector, we can plan the approach and close-to-grasp motions for the hand and fingers. While this can be done using open-loop control with pre-planned trajectories, the integration of tactile feedback from the fingertip sensors will provide a much more robust solution. We are currently implementing a two-level scheme where the approach motion is slowed down and adjusted whenever the tactile sensors indicate a hand-object contact during the early approach phase, while the fingers are still in the pre-shape pose.

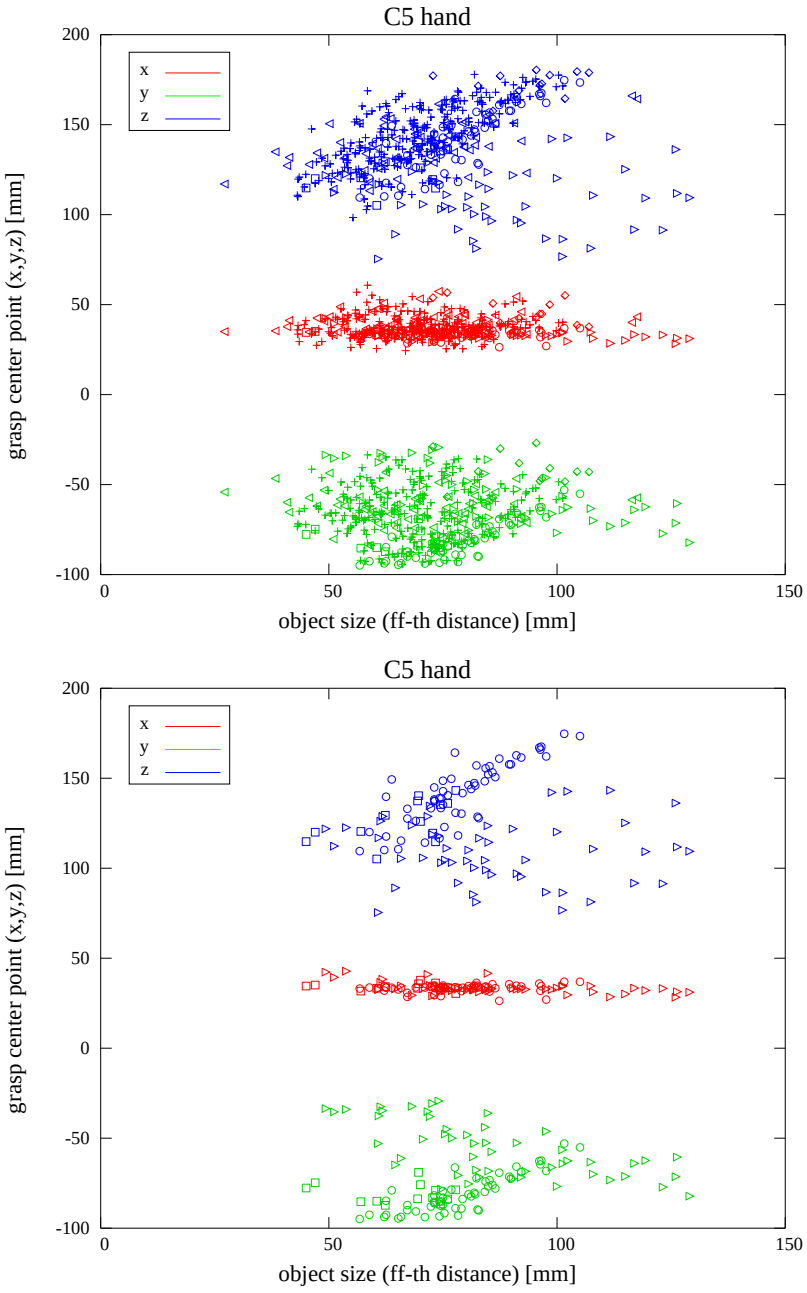

Figure 6: Estimation of the grasp origin as a function of object size. Each marker in the figure shows the estimated grasp origin for one of the human-like grasps recorded on the C5 air-muscle hand, calculated by running forward kinematics on the recorded joint-angles. top: all eight grasp types recorded; right: three grasp types (lateral, writing-tripod, parallel-extension). Colors indicate the (x,y,z) coordinates, while markers indicate the grasp-type. Note that the workspace of the hand is very small in the x-direction, but grasp origin correlates with grasp-type and object-size in the y- and z-directions.

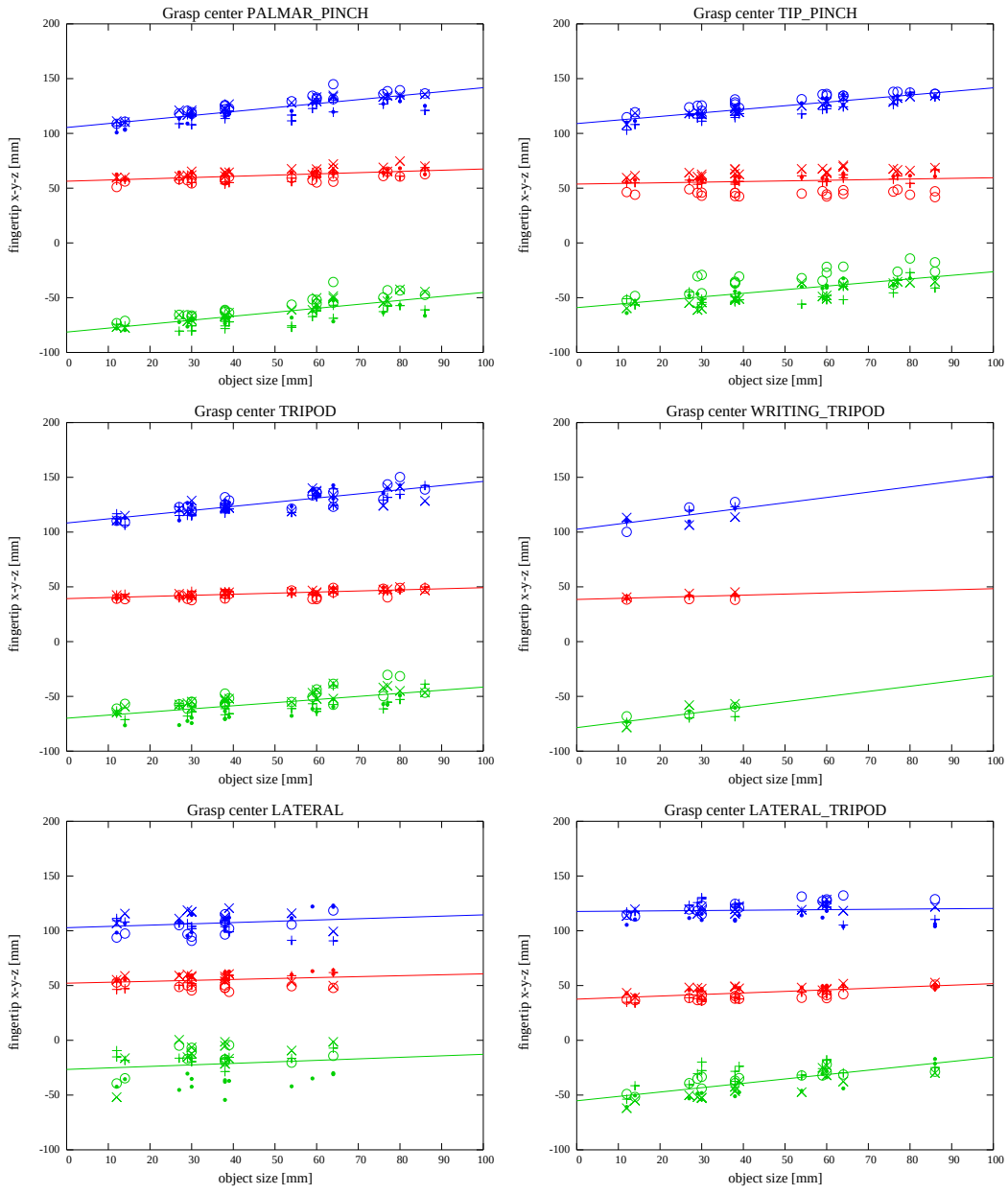

Figure 7: Estimation of the grasp center point as a function of object size and grasp-class. Colors indicate the (x,y,z) coordinates, markers indicate experimenters. From top to bottom and left to right: palmar pinch, tip pinch, tripod, writing tripod, lateral, lateral tripod.

ID	Grasp class	x_0	Δx	y_0	Δy	z_0	Δz
08	PALMAR_PINCH	56.32	0.11	-81.42	0.36	105.38	0.36
13	TRIPOD	39.27	0.10	-69.97	0.28	108.26	0.38
15	LATERAL	52.14	0.09	-26.60	0.14	102.82	0.12
19	WRITING_TRIPOD	38.49	0.10	-78.52	0.47	102.54	0.48
21	PARALLEL_EXT.	43.71	0.10	-82.29	0.32	102.17	0.40
22	ADDICTION_GRIP	21.64	0.45	-33.99	-0.43	184.01	-0.52
23	TIP_PINCH	53.70	0.06	-59.12	0.33	108.95	0.33
24	LATERAL_TRIPOD	37.54	0.14	-55.24	0.40	117.61	0.03

Table 1: Estimation of the grasp center point as a function of object size (in millimeters) and grasp type. The values are the linear least-squares approximation to the grasp-center point from human-demonstration on the Shadow C5 hand, as a function of object size for the studied eight grasp-types considered.

ID	Grasp class	(a_x, a_y, a_z)
08	PALMAR_PINCH	$(0, -1, 0)$
13	TRIPOD	$(0, -1, 0)$
15	LATERAL	$(0.71, 0, 0.71)$
19	WRITING_TRIPOD	$(0, -1, 0)$
21	PARALLEL_EXT.	$(0, -0.71, 0.71)$
22	ADDICTION_GRIP	$(0, -0.71, 0.71)$
23	TIP_PINCH	$(0, -1, 0)$
24	LATERAL_TRIPOD	$(0, -1, 0)$

Table 2: Estimated hand approach vectors $a = (a_x, a_y, a_z)$ for the studied eight grasp-classes.

In this first phase, the whole hand is moved sideways in order to compensate for slight errors in estimated object position or size, while the motion is stopped completely if multiple unexpected fingertip contacts are measured, for example due to an undetected obstacle. The second level consists of adaptive closing of the finger towards the object; fingers are slowed down or stopped on first contact until the tactile sensors of all fingers corresponding to the grasp class indicate object contact. Finally, all fingers are moved again to close around the object, or finger forces are increased as necessary in force-control mode.

3.4 Execution of manipulation motions

We have started a set of experiments to learn and demonstrate all manipulation motions of the taxonomy from Elliott and Connolly [21]. The taxonomy first considers in-hand actions like reorienting an object between thumb and index-finger, but also includes the basic finger gaiting sequences.

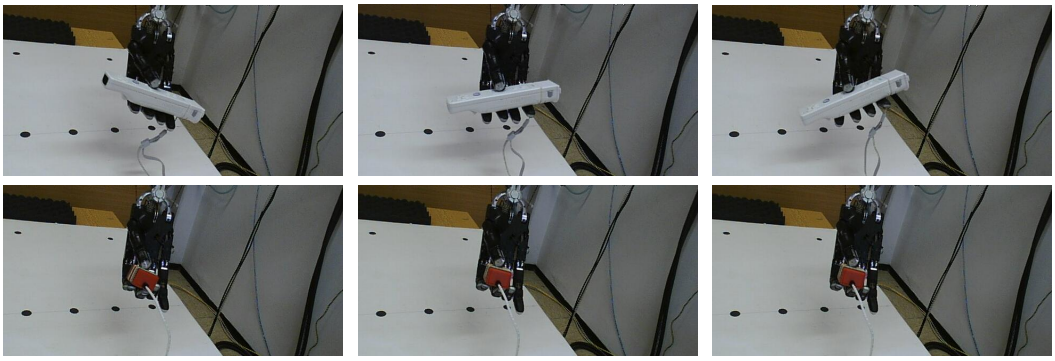

Figure 8: Above: Rotation of the Wiimote controller using parallel-extension grasp. Below: Rotation of the Joywarrior accelerometer using the tripod grasp.

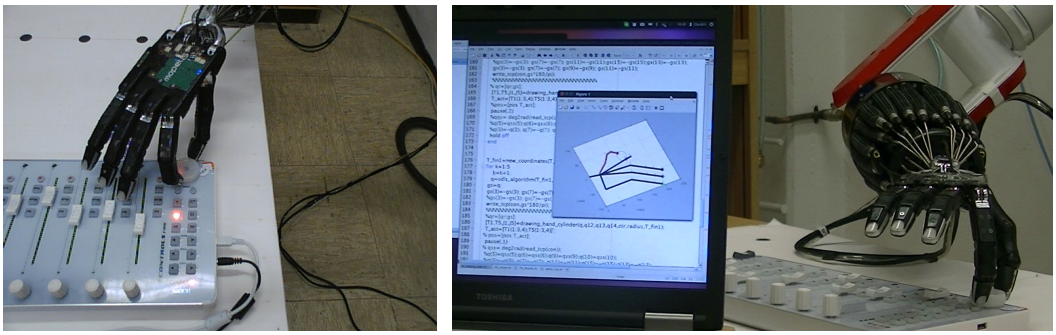

Figure 9: Rotating the jog-wheel on the iControlsPro using palmar-pinch grasp.

4 Efficient Motor Babbling for Grasping

Robot grasp planning and optimization have been hot topics in the last decade and are still under very active research. Most of the proposed approaches rely on off-line sampling strategies: candidate grasps are generated according to some criteria and then ranked with some quality metric obtained in simulation environments. Uniform sampling around heuristic pre-grasps [83], [84], simulated annealing [73], randomized trees [88] and active learning [78] are state-of-the-art techniques to generate grasp candidates. However, such approaches often assume the availability of "perfect" models of the robot kinematics, control and object geometry, which does not apply in practice. The problem is very challenging since it involves the integration of multiple robot skills: object perception, motor control, force and tactile sensing, robot kinematics and dynamics. Each of these skills introduce some sort of uncertainty either due to noise in sensors or modeling (calibration) errors. Because of such errors, deterministic approaches to grasp planning are brittle and prone to failures. Under uncertainty and systematic errors, more precise models of the grasping problem can be obtained by learning approaches. Trial and error approaches can be used to learn models of the grasping process but exploration and feedback strategies must be carefully chosen otherwise the problem may become intractable due to the large dimension of its state-space. We propose the use of Bayesian Optimization methods [72] to tackle the problem of efficient exploration. Our work exploits a sequential sampling strategy, where the results from previous trials convey information to guide the next samples. We show experimentally that, depending on an object's shape properties, sequential decision may significantly reduce the number of trials necessary to achieve quasi-optimal grasps with respect to random search. Despite being recently used in machine learning applications such as learning robot parameters [81], learning neural network weights [76], finding policies for robot path planning [82], etc, we introduce Bayesian Optimization methods to the robot grasping problem.

We consider robot hands with the ability to detect forces and contact points with objects. Most recent robotic hands have some sort of tactile or force/torque sensing allowing this ability. For instance, the Shadow Robot Hand has been recently installed with force/torque sensors in each fingertip in a special encapsulation (see Fig. 10 on page 24) that allows the reconstruction of the contact points and force information [79]. In these cases grasps can be evaluated by a quality metric through the analysis of the force/torque pairs (wrenches [85]) applied by the hand on the object. Several of these metrics have been proposed in the literature, often using some form of wrench space analysis. The one we use is based on [87] and provides quality measures for both force-closure [85] and non-force-closure grasps. Force-closure grasp quality depends on the highest magnitude wrench that the grasp can resist in any direction. For non-force-closure grasps the quality depends on how distant the grasp is to being force-closure. The evaluation of non-force-closure grasps is not common in grasp planning literature but we find the additional information

Figure 10: The Shadow Robot Hand equiped with fingertip force/torque sensors ATI NANO 17.

obtained from non-force-closure grasp assessment to be extremely useful. The proposed method consists of performing consecutive grasp trials, changing the grasp parameters x (for instance hand-object relative pose, hand closure strategy, etc) until good grasps are achieved. Each grasp trial should use the information obtained in the previous ones as much as possible in order to minimize the number of trials to reach a good grasp. The objective is to optimize a set of grasp parameters to obtain the highest quality grasp metric value.

4.1 Quality Evaluation Metric

The modern approach to robotic grasping tries to understand and emulate how humans use their hands to explore, restrain and manipulate objects. On account of the human hand complexity which provides extensive sensory feedback (sensing slip, object weight, object stability, etc.), humans intuitively evaluate the quality of the grasp they are performing. Robots on the other hand have manipulator and sensor limitations that increase the difficulty of the grasping process. A quality metric for grasp assessment is required as an interpretation of the sensory feedback given by the robot manipulator. Measuring the quality of stable grasps but also giving a measure of the distance to stability of non-stable grasps, while transitioning from stable to non-stable grasps in a smooth fashion, are the core aspects aimed at when designing a grasp quality metric.

Assuring that the metric is designed to fulfill the above requirements is not enough do ensure precise grasp assessment. As the metric's inputs, the contact points are

crucial to the performance of the metric and achieving a realistic representation of these contacts leads to significant error reduction and more accurate results.

Here we will depict how it is possible to attain a fair reproduction of what is contemplated on the real system by using a realistic model of the contact points in conjunction with a bimodal wrench space analysis metric.

4.1.1 Wrenches

A grasp is no more than a set of forces applied on a rigid body by a manipulator. Each of these forces consists of a linear component (pure force) and an angular component (pure moment) acting at a point. Representing this force/moment pair as a vector in $\mathbb{R}^6$ defines a wrench w, [85].

$$w = \begin{bmatrix} f \\ \tau \end{bmatrix} \quad \begin{matrix} f \in \mathbb{R}^3 \\ \tau \in \mathbb{R}^3 \end{matrix} \tag{1}$$

The values of this wrench vector $w \in \mathbb{R}^6$ depend on the coordinate frame in which the force and the moment are represented. If B is a coordinate frame attached to a rigid body, then $w_b = (f_b, \tau_b)$ is the wrench applied at the origin of B, with f_b and τ_b specified with respect to the B coordinate frame.

If several wrenches are applied on the same rigid body the resulting net wrench can be constructed by adding all the wrench vectors. This addition only makes sense if all the wrench vectors are represented with respect to the same coordinate frame. Thus, if all wrenches in a wrench set w_i are to be added, all the wrench vectors must be rewritten to a singe coordinate frame before the addition is performed. If B and C are distinct coordinate frames, the transformation of a wrench w_b applied at the origin of the B frame to an equivalent wrench w_c applied at the origin of the C frame can be achieved by

$$\begin{bmatrix} f_c \\ \tau_c \end{bmatrix} = \begin{bmatrix} R_{cb} & 0 \\ [p_{bc}]_\times R_{cb} & R_{cb} \end{bmatrix} \begin{bmatrix} f_b \\ \tau_b \end{bmatrix}. \tag{2}$$

where $[p_{bc}]_\times$ is the skew-symmetric matrix of p_{bc} defined by

$$[p]_\times = \begin{bmatrix} 0 & -p_3 & p_2 \\ p_3 & 0 & -p_1 \\ -p_2 & p_1 & 0 \end{bmatrix} \tag{3}$$

where p_1, p_2 and p_3 are the components of vector p.

With the use of this matrix the transformation includes an additional torque $p_{bc} \times f_c$ which is the torque generated by applying the force f_b at p_{bc}.

4.1.2 Contact Model

Using wrenches as a representation of forces and torques, it now becomes necessary to model the contact between the object and the manipulator. This model will provide the wrenches produced at each contact point. Each contact point is represented by a coordinate frame, R_{c_i}, attached to the contact location, p_{c_i}, which is defined by its relative position and orientation with respect to a reference frame R_r. For example, the coordinate frame centered on the object's center of mass. The coordinate frame R_{c_i} is chosen such that its z-axis points in the direction of the surface normal at the point of contact.

Several models for these contact points have been considered. The frictionless point contact model, [85], is the simplest of the considered models. In this model no friction between the contacting surfaces is taken into account. Thus the model only allows forces along the surface normal direction. Using this model the produced wrenches at each contact c_i can be represented as

$$w_{c_i} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} f_{c_i}, \quad f_{c_i} \geq 0 \tag{4}$$

where $f_{c_i} \in \mathbb{R}$ is the magnitude of the applied normal force.

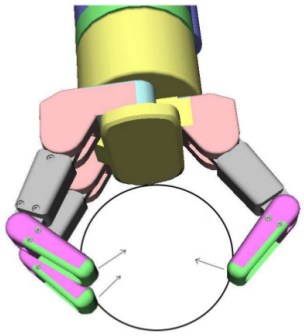

Figure 11: Frictionless point contact model.

Since not considering friction is a fairly naive approach, a friction model has to be introduced. The friction model chosen is Coulomb's friction model, [85]. This model states that forces in the tangential direction to the contact surface can be introduced with maximum magnitude proportional to the magnitude of the normal

component. The constant of proportionality between the two components is the static coefficient of friction μ_f

$$\mathsf{f\^{}\{t\}} \leq \mu_f \mathsf{f\^{}\{n\}} \tag{5}$$

where $f^t \in \mathbb{R}$ and $f^n \in \mathbb{R}$ are the magnitudes of the tangential and normal force components respectively. The geometric meaning of this condition is that any applied force has to lie inside a cone centered about the surface normal at the point of contact. This cone is called the friction cone. Using Coulomb's model, it is a simple task to derive the point contact with friction model.

$$w_{c_i} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} f_{c_i}, \quad f_{c_i} \in FC_{c_i} \tag{6}$$

where the friction cone FC_{c_i} is

$$FC_{c_i} = \{f \in \mathbb{R}^3 : \sqrt{f_1^2 + f_2^2} \leq \mu_f f_3, f_3 \geq 0\}. \tag{7}$$

The components f_1, f_2 and f_3 represent the force along the x, y and z coordinates respectively.

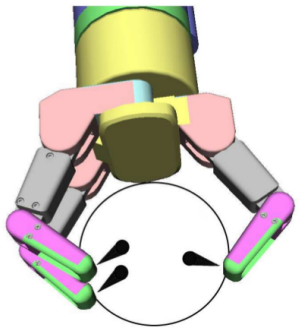

Figure 12: Point contact with friction model.

The soft-finger contact model, [85], is a more complete model that allows forces to be applied in a cone about the surface normal but also allows torques about that

normal. Using this model the resulting wrench at each contact point is

$$w_{c_i} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} f_{c_i}, \quad f_{c_i} \in FC_{c_i} \tag{8}$$

where the friction cone FC_{c_i} is defined as

$$FC_{c_i} = \{f \in \mathbb{R}^4 : \sqrt{f_1^2 + f_2^2} \leq \mu_f f_3, f_3 \geq 0, \mathsf{f\_\{4\}} \geq \gamma \mathsf{f\_\{3\}}\} \tag{9}$$

with γ representing the torsional friction coefficient and f_4 the torque magnitude along the contact normal direction. This is the chosen model for this work since it is the most realistic model of the ones presented previously.

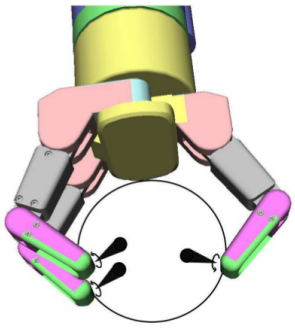

Figure 13: Soft-finger contact model.

While analyzing a grasp, the normal force is assumed to be of unit magnitude

$$f_3 = 1 \tag{10}$$

and the friction cone is sampled over its outer limits. The resulting set of wrenches at each contact point

$$w_{c_i}^k = \begin{bmatrix} \mu_f sin(\theta_k) \\ \mu_f cos(\theta_k) \\ 1 \\ 0 \\ 0 \\ \pm\gamma \end{bmatrix}, \quad \theta_k \in [0, 2\pi] \tag{11}$$

offer the base framework for grasp formulation.

One important aspect of this framework is that although the common choice for the reference frame R_r is the coordinate frame centered on the object's center of mass, the end results will be independent of the chosen reference frame, as will be shown in the experimental results. This allows the use of this framework on non-model based systems as long as, when comparing two grasps, both of them are represented with respect to the same reference.

4.1.3 Contact Surface Model

Using a single-point contact model might not be a completely realistic approach for contact modeling when using anthropomorphic grippers, since the gripper surfaces tend to mold slightly to the object surface. Single-point contact models also have a low tolerance to errors, especially errors of the contact normal direction. Taking this into account, a more realistic approach would be to employ a surface contact model which is not only more robust to errors but can also provide a better compliance with the object's surface geometry.

The first step is to preprocess the raw contact point cloud given by a physics engine in simulation or by a set of sensors on a real system. By simply calculating the mean of the raw contact point cloud and defining a single-point contact with the mean position, p_m, and the mean normal direction this preprocessing is achieved in a simple and fast manner. To represent the mean normal direction, a new coordinate frame is created, R_m, with z axis along the mean normal direction and origin p_m.

$$c_m = (p_m, R_m) \tag{12}$$

Afterwards, a series of line segments parallel to the contact normal of c_m are generated. These line segments are bounded by a sphere centered in p_m with radius r and by a plane containing p_m and orthogonal to the contact normal of c_m. Calculating the intersections of each of the line segments with the object surface the points that define the contact surface are found. If multiple intersections are found for the same line segment the one closest to p_m is chosen. The final step is to calculate the object surface normals of the points spanning the contact surface. The points in the contact surface are then modeled with the point-contact model described previously, each of them generating a set of wrenches $w_{c_i}^k$. Fig. 14 depicts the procedure described above.

4.1.4 Grasp Representation

With the robust contact model framework introduced, defining a grasp is straightforward. Since all contacts are defined as sets of wrenches it is necessary to transform them to a common reference frame R_r. The set of all the transformed wrenches

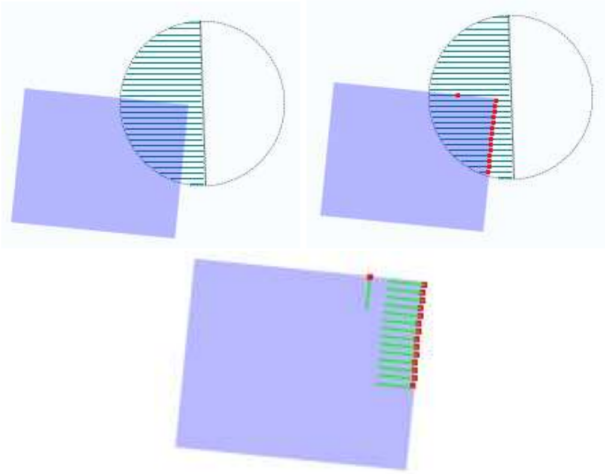

Figure 14: Steps for generating the Surface Contact Model. Line segments generated (top left), line segments and their intersections with the object surface (top right) and intersection points and their correspondig surface normals (bottom). The blue box represents the object, the red points represent the intersections between the line segments and the object and the light green lines represent the object surface normal at each intersection point.

defines the grasp, and is designated as grasp map G, [85]. Assuming that n contacts are generated

$$G_i = \begin{bmatrix} R_{c_i} & 0 \\ [p_{c_i}]_\times R_{c_i} & R_{c_i} \end{bmatrix} w_{c_i} \quad i \in [1, ..n] \,. \tag{13}$$

and the resulting grasp map is

$$G = [G_1, ..., G_n] \,. \tag{14}$$

4.1.5 Closure

Force-Closure is a binary qualitative evaluation of grasp stability, [85]. If a grasp can resist any applied wrench it is considered force-closure. In other words given an external wrench $w_e \in \mathbb{R}^6$ applied to the object, there is a combination of contact forces f_c such that

$$G f_c = -w_e \tag{15}$$

where

$$f_c = \begin{bmatrix} f_{c_1} \\ \vdots \\ f_{c_n} \end{bmatrix}, \quad f_{c_i} \in FC \tag{16}$$

and n is the number of contacts that compose the grasp. A simple way to evaluate if a grasp G is force-closure is given through the analysis of $ConvexHull(G)$. The $ConvexHull(G)$ represents the minimum convex region spanned by G on the wrench space W. Defining the grasp wrench space, W_G, as the space of all possible wrenches generated by the grasp

$$W_G = ConvexHull(G) \tag{17}$$

then G is force-closure if

$$W_0 \subset W_G \tag{18}$$

where W_0 is a small neighborhood of the wrench space origin.

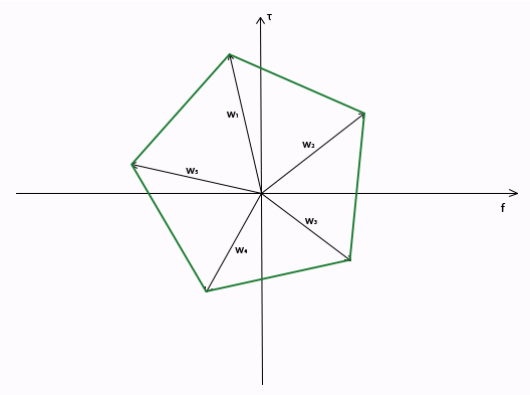

Figure 15: Convex Hull of the grasp wrenchs, the Grasp Wrench Space

4.1.6 Bimodal Wrench Space Analysis

Typical metrics can give a measure of how good a force-closure grasp is but for non-force-closure grasps they give no information. Since for the set of all possible grasps that can be executed on a given object, only a small portion will pass the force-closure test, there is a large portion of the possible grasp set that remains unevaluated. The proposed metric attempts to solve this problem by evaluating all grasps even if they are non-force-closure, by means of different analysis modes chosen in accordance to the results of a force-closure test. Thus, for any configuration of the robot manipulator that touches the object (even if only with one finger) the metric will return a measure of grasp quality.

The proposed metric has two distinct analysis modes (one for force-closure and one for non-force-closure grasps) hence is being called the Bimodal Wrench Space Analysis Metric (BW). The first mode, used for force-closure grasps, measures the radius of the largest sphere centered on the origin of the wrench space W that is contained in W_G. Conceptually it represents the magnitude of the largest external force that can be resisted by the grasp in any direction. Because all the analysis is

done under the assumption of unit normal force at all contact points, grasps where the sphere radius is larger have increased stability with the same amount of applied force.

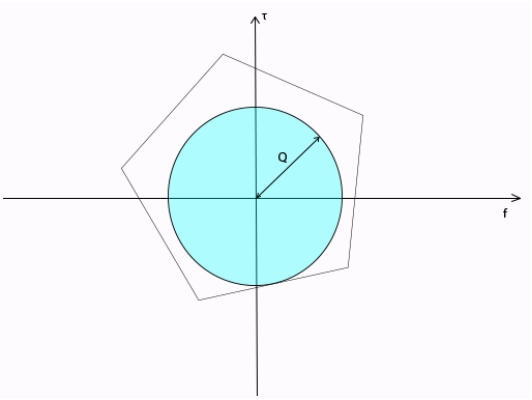

Figure 16: The Largest Sphere metric. Used by the BW metric to analyse force-closure grasps.

The second mode, used for non-force-closure grasps, measures the minimum distance from the origin of the wrench space W to a point contained in the W_G

$$min_y : y \subset W_G \tag{19}$$

Knowing that for a grasp to be force-closure the condition in (18) has to be met, it can be reasoned that, for non-force-closure grasps, if the grasp wrench space for grasp A, W_{G_A}, is closer to the origin than the grasp wrench space for grasp B, W_{G_B}, the changes in G_A in order to reach force-closure are inferior to the changes in G_B to reach the same condition. This is the reasoning behind the measure given by the second mode. In other words, mode two measures the distance to force-closure of a non-force-closure grasp.

A computationally interesting point of this second mode is that by representing the grasp wrench space region as

$$Nx \leq b \tag{20}$$

where N are the normal vectors and b the offsets that as a pair define each of the planes containing W_G, the second analysis mode can be represented as a simple convex optimization problem that can be solved easily by convex optimization solvers.

$$min_x : Ax \leq b \tag{21}$$

To ensure smooth transitions between the two modes and hence a continuous metric function, the symmetric value given by mode two is used. This makes sense because as the distance to stability grows larger the overall evaluation of the grasp should be worse. On the other hand when the distance to stability tends to zero, the grasp is near transitioning to a force-closure and to positive metric values.

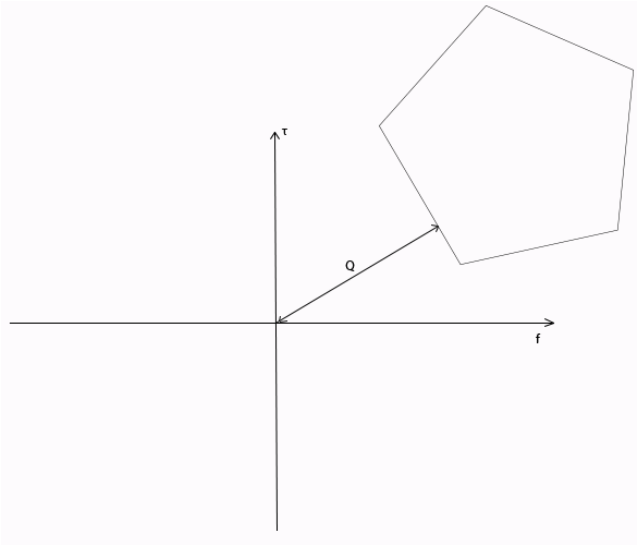

Figure 17: The BW metric. The grasp wrench space distance to force-closure when analysing a non-force-closure grasp.

4.2 Bayesian Optimization

One of the key points of our method is to define a suitable exploration strategy able to minimize the number of required exploration trials to achieve good grasps. To address this issue we will employ recent results in global sequential optimization using Bayesian methods [72]. The grasp quality metric will be maximized through a search strategy that proposes new trials on regions of the parameter space where either its expected value (exploitation) or its uncertainty (exploration) are large. The outline of the proposed method is as follows:

An initial configuration of arbitrary grasp parameters x_0 (position and pose with respect to object, sinergies, etc) is defined for the first robot trial. This can be defined in several ways: either randomly on a bounding box around the object or using heuristics from prior knowledge. This robot grasp trial feeds a Gaussian Process Regression model [86] that models the expected value and variance of the quality metric. These measures can then be predicted easily at any point in the state space. To decide the next point to try, we maximize at each time step a form of Expected Improvement function (EI) with exploitation-vs-exploration control [80]. To optimize the expected improvement function we use the DIRECT algorithm [75]. This is a global optimization method that works by partitioning the space in intervals (DIRECT stands for DIvide RECTangles), estimating the Expected Improvement (EI) function in their center, and choosing, at each time step, the interval where EI can be maximal for any bound on the function derivative (Lipschitz constant). The obtained solution will define the parameters of the next robot trial. This cycle is depicted in Fig. 18.

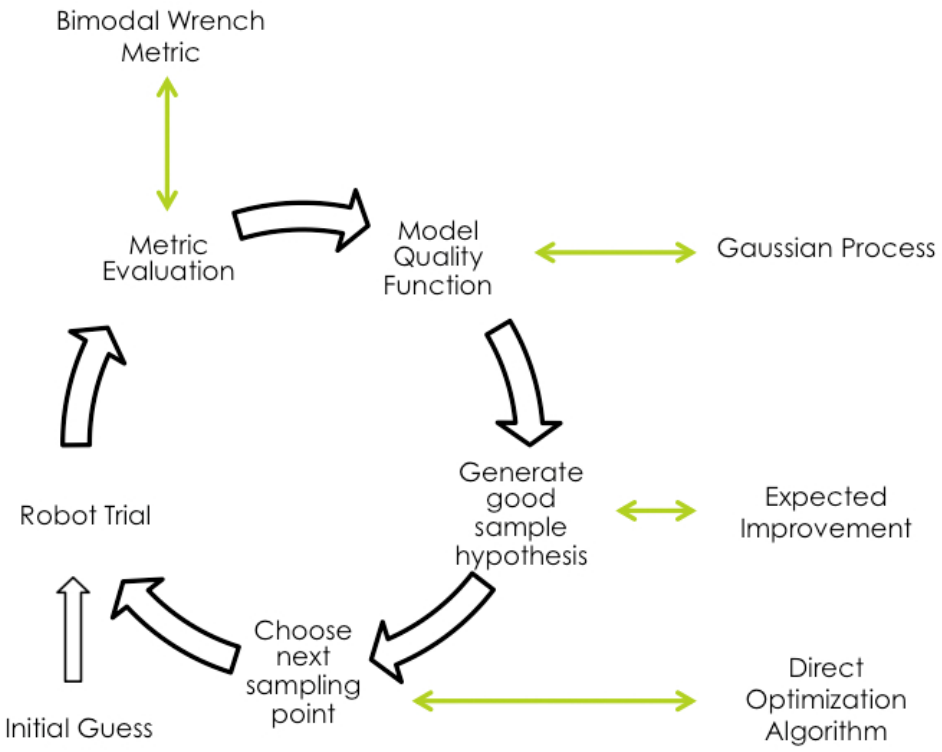

Figure 18: Complete method diagram.

4.2.1 Gaussian Process Regression

A Gaussian process allows the information about the subject of interest to be represented in a way that the learning agent understands. It is used as a means to describe a distribution over functions. As a more formal definition, a Gaussian Process (GP) is a collection of random variables, any finite number of which have a joint Gaussian distribution, [86]. It is completely defined by a mean and covariance function pair

$$\mu(x) = \mathbb{E}\left[f(x)\right] \tag{22}$$

$$\Sigma(x, x') = \mathbb{E}\left[(f(x) - \mu(x))(f(x') - \mu(x'))\right] \tag{23}$$

and is represented as

$$f(x) \sim GP(\mu(x), \Sigma(x, x')). \tag{24}$$

In the current context the random values represent values from the grasp quality metric function $f(x)$ that the robot wishes to learn. The mean function is the best prediction of the true function given what is known. Also it is initially considered as a zero function since there is no relevant information at the start of the process, but other priors may be used if desired (e.g. metrics learned with other objects). The covariance function is what gives a sense of proximity or similarity between two points.

A kernel is the general name given to a function K of two arguments mapping a pair of inputs $x \in X$, $x' \in X$ into $\mathbb{R}$. In this work the kernel chosen is a Matérn class covariance function given by

$$K(r) = (1 + \frac{\sqrt{3}r}{l})exp(-\frac{\sqrt{3}r}{l}) \tag{25}$$

where

$$r = \mathsf{x} - \mathsf{x}' \tag{26}$$

measures the distance between points x and $x'$and l represents the kernel length. Resorting to kernel functions allows us to create covariance functions where the connectivity between points is directly related with the distance between them. For more information on kernel functions refer to [86].

The representation of a covariance function is what implies a distribution over functions. Our goal is to perform the estimation of $f(x)$ based on the already known function values. To do this it is a simple matter of conditioning the distribution over functions to what is already known

$$F_* \mid X_*, X, F \sim N(\mu(x), \Sigma(x, x')) \tag{27}$$

$$\mu(x) = K(X_*, X)K(X, X)^{-1}F \tag{28}$$

$$\Sigma(x, x') = K(X_*, X_*) - K(X_*, X)K(X, X)^{-1}K(X, X_*) \tag{29}$$

where X_* is the estimation point set, X is the observation (known) point set, F is the set of observed function values and F_* denotes the estimated function values.

The values $K(X_*, X)$, $K(X, X_*)$, $K(X, X)$ and $K(X_*, X_*)$ are the kernel evaluated between point pairs of the prediction and observation set.

Despite the fact that this is a fairly good estimation of the function it is still somewhat naive. The thought of getting perfect measurements experimentally is an extremely gullible approach. Therefore some observational error has to be taken into account when preforming the estimation. Assuming additive independent identically distributed Gaussian noise ε with variance σ_n^2 leads to

$$F_*X_*, X, F \sim N(\mu(x), \Sigma(x, x')) \tag{30}$$

$$\mu(x) = K(X_*, X)(K(X, X) + \sigma_n^2 I)^{-1}F \tag{31}$$

$$\Sigma(x, x') = K(X_*, X_*) - K(X_*, X)(K(X, X) + \sigma_n^2 I)^{-1}K(X, X_*) \tag{32}$$

where I is the identity matrix.

This estimation of the mean function provides us with the robust estimation of the real function that we need. It also provides a measure of the estimation's uncertainty, which will be crucial to the robot's learning.

4.2.2 Expected Improvement

Now that we have something that provides a solid representation of what we wish to learn and that allows us to predict what is still unknown based on current knowledge, the next step is to decide what we should do to improve this knowledge. This decision should not be taken lightly since it is what defines the rate at which we learn. To achieve this decision we will use the concept of Expected Improvement [72].

The Gaussian Process provides a global estimation of $f(x)$ based on what is known at the time. Since we are trying to find the maximum value of the function $f(x)$ the natural decision should be to explore regions of the function where a higher maximum is possible. From this idea we define the improvement function as

$$I(x) = \max\{0, f_{n+1}(x) - f^{max}\} \tag{33}$$

where f^{max} is the current maximum value. The function takes on positive values when the prediction is higher than the best value found so far and is set to zero otherwise. Using the improvement function, the new observation point is attained by finding the maximum expected improvement point

$$x = \arg\max_{x} \mathbb{E}(\max\{0, f_{n+1}(x) - f^{max}\} \mid \boldsymbol{\mathcal{D}}_n) \tag{34}$$

where $\mathcal{D}_n$ is all that is known at time n

$$\mathcal{D}_n = [f_1, \ldots, f_n] \tag{35}$$

and f_i is the observed value for trial i. This expected improvement can easily be evaluated analytically, [77], through

$$EI(x) = \begin{cases} (\mu(x) - f^{max})\Phi(Z) + \sqrt{\Sigma}\phi(Z) & if\ \Sigma > 0 \\ 0 & if\ \Sigma = 0 \end{cases} \tag{36}$$

where

$$Z = \begin{cases} \frac{\mu(x) - f^{max}}{\sqrt{\Sigma}} & if\ \Sigma > 0 \\ 0 & if\ \Sigma = 0 \end{cases}, \tag{37}$$

Σ is the covariance function and $\phi(.)$ and $\Phi(.)$ respectively denote the probability density function and the cumulative distribution function of the standard Normal distribution. As mentioned in the previous section, the uncertainty measure given by the covariance function Σ plays a huge role in the decision of the next observation. It enables the balancing between exploiting and exploring. When exploring, we should focus on points where the prediction variance is large in order to minimize the global uncertainty. When exploiting we should focus on the points where the

predicted mean function is high so that a higher and more accurate value of the global maximum may be found.

Evaluating the expected improvement through (36) balances the exploration versus exploitation trade-off in an unruly fashion. A more generalized form of the $EI(.)$ has to be found in order to directly control this balance. Lizotte [80] suggests a $\xi \geq 0$ parameter obtaining

$$EI_{\xi}(x) = \begin{cases} (\mu(x) - (f^{max} + \xi))\Phi(Z_{\xi}) + \sqrt{\Sigma}\phi(Z_{\xi}) & if\ \Sigma > 0 \\ 0 & if\ \Sigma = 0 \end{cases} \tag{38}$$

where

$$Z_{\xi} = \begin{cases} \frac{\mu(x) - (f^{max} + \xi)}{\sqrt{\Sigma}} & if\ \Sigma > 0 \\ 0 & if\ \Sigma = 0 \end{cases}. \tag{39}$$

In this work we use $\xi = \frac{\hat{\sigma}_f^2}{100}$ where $\hat{\sigma}_f^2$ is the Gaussian process estimated variance given by

$$\hat{\sigma}_f^2 = F^T K(X, X)^{-1} F. \tag{40}$$

For a more detailed reading on this topic refer to [72] and [80].

As it now stands, the method seems complete. We have a procedure to represent the current knowledge, to predict the unknown and to accordingly decide what to do next. Repeating this process will ultimately lead to learning $f(x)$ very efficiently in terms of minimizing the number of observations. In the next section we will show that one more aspect must be considered for the method to be efficient.

4.2.3 Direct Optimization Algorithm

Now that the method is fairly complete, one question arises. Is it computationally feasible? The answer is yes, struggling for higher dimensional inputs. To optimize the Expected Improvement function one needs to evaluate it at several points. Kernels must be evaluated at all point pairs possible between the points in the estimated and observation point sets. While the kernel evaluation is simple, the number of points in the estimated point set grows at troubling rates if uniformly sampled through input space. This point set grows at a rate of n^D where n is the number of samples per dimension and D is the number of input parameters. So a simple uniform sampling of the space along all the dimensions in order to find the point with the highest expected improvement is not a good approach.

We turn to a more efficient approach trough the use of the Direct Optimization algorithm, [75]. This algorithm uses a small number of initial predictions to decide how to DIvide the feasible space into smaller RECTangles. The end result is a high discretization of the target function near the function maxima and a low discretization elsewhere.

The Direct algorithm starts by normalizing the function domain into a unit hyper-cube with center c_1

$$\bar{\Omega} = \{x \in \mathbb{R}^N : 0 \leq x \leq 1\}. \tag{41}$$

The algorithm works in this normalized space, only reverting to the original space when making function calls. After evaluating the function at $f(c_1)$ it is time to make the first division of the hyper-cube. The cube is divided into smaller cubes centered at $c_1 \pm \delta e_i$, $i = 1, ..., n$ where δ is one third of the cube length and e_i is the ith unit vector. Direct choses to leave the best function values in the largest space. As such the first dimension to be divided is chosen by means of

$$\omega_i = \min(f(c_i + \delta e_i), f(c_i - \delta e_i)), \quad 1 \leq i \leq N. \tag{42}$$

The dimension with the smallest ω_i is divided into thirds and the process is repeated for all dimensions on the resulting center hyper-rectangles.

With the hyper-cube division done it is time to find which of the newly generated rectangles/cubes may be potentially optimal. For the optimality test, we test each of the rectangles/cubes for the existence of a Lipschitz Constant $\hat{K} > 0$ that allows

$$f(c_j) - \hat{K}d_j \leq f(c_i) - \hat{K}d_i, \forall i, \tag{43}$$

$$f(c_j) - \hat{K}d_j \leq f_{min} - \epsilon \mathsf{f\_\{min\}} \tag{44}$$

where $\epsilon > 0$ is a positive constant, f_{min} is the current best function value and c_j and d_j are respectively the center of the tested hyper-rectangle/cube and a measure of the dimension of the same rectangle/cube. In (43) we test if the possible variation of the value $f(c_j)$ when traveling inside the respective hyper-rectangle/cube may reach a minimum value when applying the same variation scale to all other $f(c_i)$ navigating on the respective ith rectangle/cube. On the other hand (44) tests if the possible minimum reached on the jth rectangle/cube is of interest when compared with f_{min}. The term $\epsilon \mathsf{f\_\{min\}}$ makes sure that the improvement to the minimum value is non-trivial.

Now that we know S, the set of all the potentially optimal rectangles/cubes, the only remaining step is to divide all members of this set, evaluate the function at the center of the resulting rectangles/cubes and update f_{min}. An interesting fact is that when dividing a hyper-rectangle, the algorithm always chooses to divide along the longest dimension(s) to ensure that the rectangles shrink on every dimension.

The Direct algorithm repeats the previous operations (except the initialization) until S is empty, meaning no more divisions are of interest. When this stage is reached the f_{min} is the global function minimum. Since we are looking to maximize the expected improvement function, it is only a matter of supplying Direct with negative values of $EI(.)$.

4.3 Experimental Results

4.3.1 Experimental Setup

Most of the experiments performed during the course of this work were done in a simulation environment through the OpenRave simulator [74]. The environment consisted of a manipulator arm, the Barrett hand, and a few objects as shown in Figures 19 and 20.

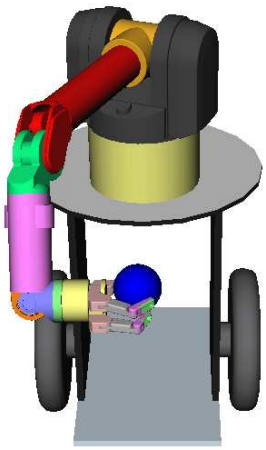

Figure 19: The experimental setup used for validating the proposed metric.

Figure 20: The objects used for validating the proposed metric.

Recently we were able to execute motor babbling in the Shadow Hand. Some preliminary results of babbling with anatomical synergies will also be presented.

4.3.2 1D exploration with Bayesian Optimization

We will now show some 1D scans made by changing the grasp's initial position along the approach axis inside a bounded region near the object. The goal of these scans is to give a better understanding of how the Gaussian process and the Bayesian methods interact. Also to be shown is how the system behaves when the initial random sample is one of the best possible or one of the worst possible. Fig. 21 depicts the latter. It represents the sequence carried out by the method to sample

a sphere, when the initial random sample does not even touch the object (top left plot). In red we can see the GP mean function which is the best estimation of the metric function at this time, the dashed lines depict the estimation variance at each function point, the red dots are the values collected from the robot trials. The blue function represents the EI function that classifies the function space in terms of exploration interest. Even with the setback caused by the bad initial random sample, it only takes 2 iterations of the method in order to find a possible maximum (top right plot). Acknowledging the fact that a region that may contain the maximum has been found, the system focuses the search in its neighboring points (bottom left plot). Once this region has been exploited, the system resumes its exploration efforts to assure there are no more regions that may contain better values of $f(x)$. After confirming that it has found the function maximum value the exploration stops (bottom right plot).

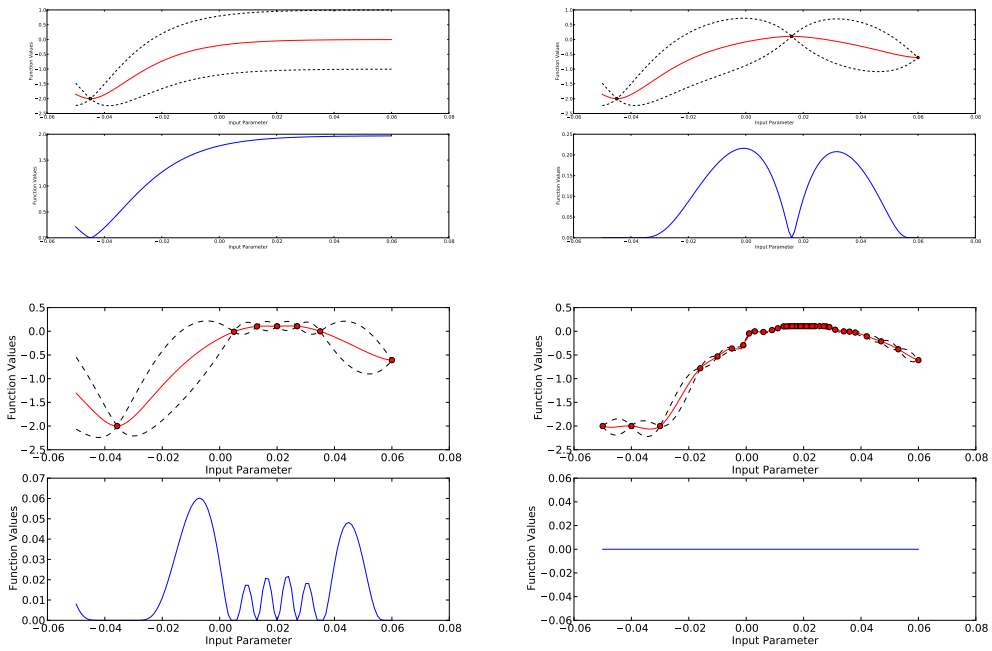

Figure 21: 1D scan of a sphere, computed by sampling along the input parameter.

The second case that will be shown is the opposite case. Fig. 22 shows the sequence followed by the method while sampling a star prism when the initial random sample is one possible maximum (top left plot). As the function is still completely unknown to the system, excluding the random sample, it is impossible for it to realize that the first sample is actually a possible maximum and so it proceeds with the exploration. After 3 iterations the system finally realizes the potential of the first sample (top right plot) examining the neighboring region (bottom left plot). The exploration is resumed and 2 more interest areas are found before the systems stops (bottom right plot).

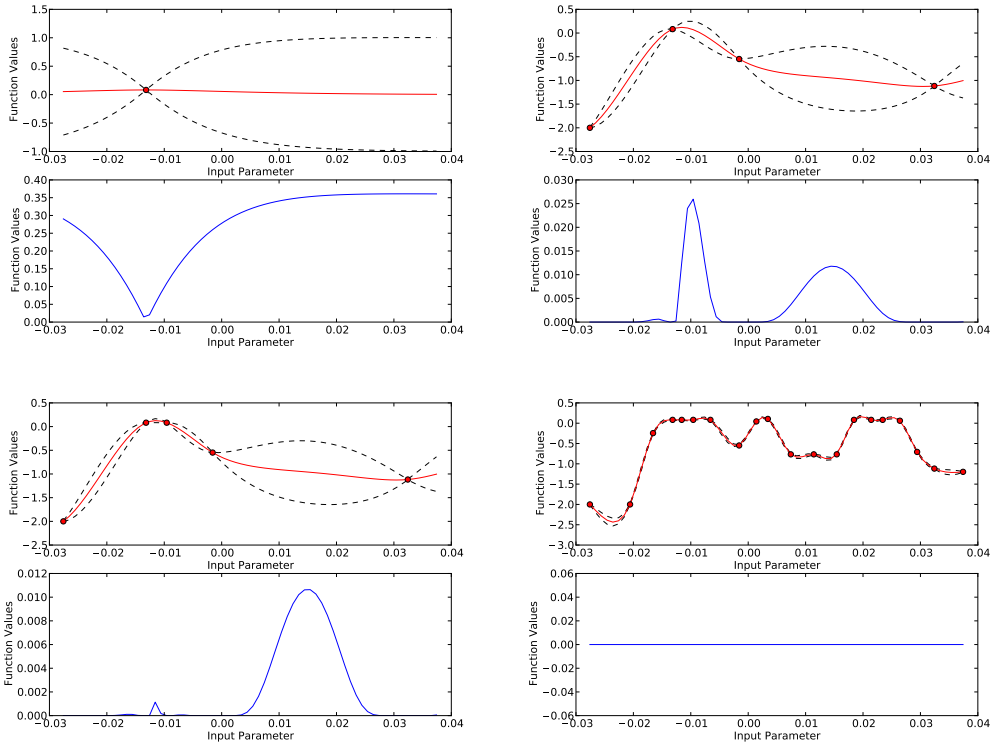

Figure 22: 1D scan of a star prism, computed by sampling along the input parameter.

4.3.3 2D exploration with Bayesian Optimization

Let us now scan the object in the 2-dimensional space. The results are shown in Fig. 23 to 25. Also in order to test the method performance under the worst possible conditions, we assume that the observations are noisy.

As shown in Tab. 3, it is clear that the number of necessary trials is different depending on the object. Also depicted in Tab. 3 are the number of trials needed in order to find a value that differs from the global maximum by less than 5%, 10% and 20% respectively from left to right.

Although only stopping when the expected improvement no longer displays interesting values, the method could have settled for much less samples and still provide a fairly good approximation of the global maximum.

4.3.4 Bayesian optimization versus random sampling

We now make a base comparison between the performance of the Bayesian optimization method and random sampling on parameter space. This comparison is

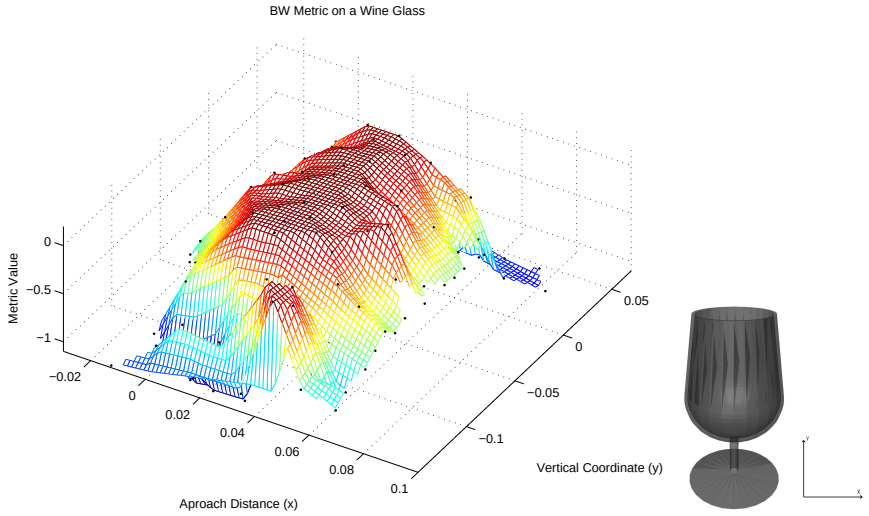

Figure 23: BW Metric values of grasps in a wine glass, computed using the Bayesian Optimization method.

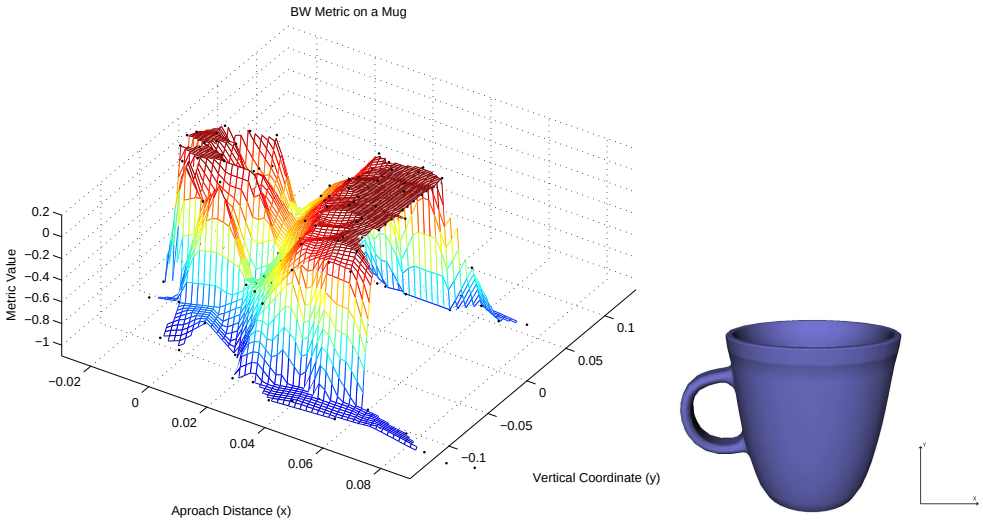

Figure 24: BW Metric values of grasps in a mug, computed using the Bayesian Optimization method.

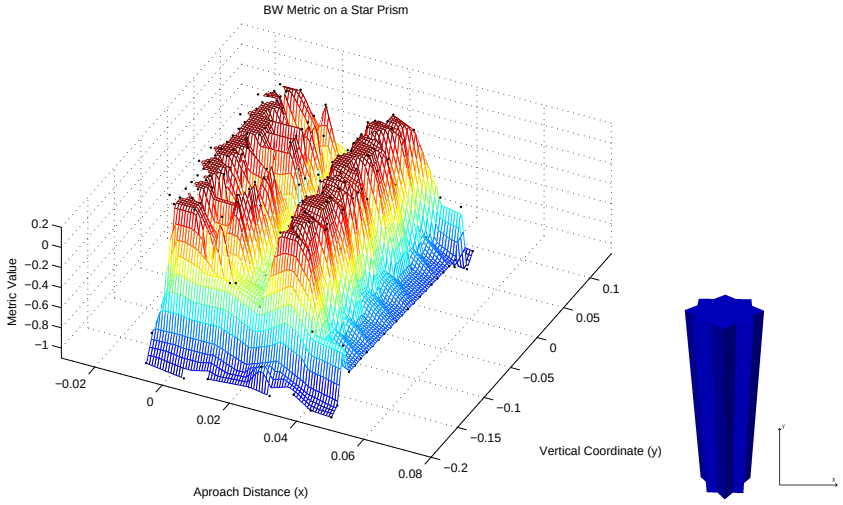

Figure 25: BW Metric values of grasps in a star prism, computed using the Bayesian Optimization method.

Table 3: Number of Samples taken for each object.

Object	**Bayesian Optimisation**			
-	Total	5%	10%	20%
Sphere	551	14	14	14
Wine Glass	306	31	19	19
Cylinder	485	39	20	20
Mug	331	55	55	1
Cuboid	564	1	1	1
Rotated Cuboid	388	59	59	59
Star Prism	402	105	105	31

done by measuring the evolution of the best value found by each algorithm along a sequence with 100 iterations when performing 2D sampling.

Fig. 26 represents the evolution of the best value found by each method when sampling as sphere. The Bayesian method (black) clearly converges faster than the random sampling method (red) to the maximum function value.

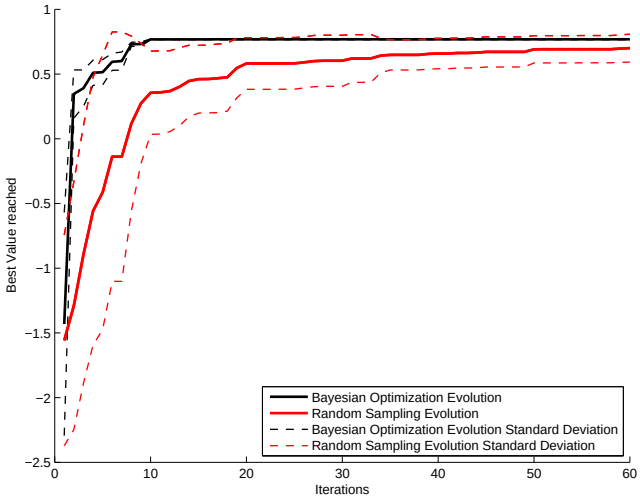

Figure 26: Comparing the evolution of best value found when sampling a sphere with the Bayesian approach versus the random sampling method.

To confirm the results obtained by the previous experiment, similar experiments were made using a mug instead of the sphere. Fig. 27 shows that although the Bayesian method is not as fast as for the sphere, it still converges faster than random exploration.

We conclude that the Bayesian method converges faster to the maximum value of $f(x)$ when compared with the random sampling method.

4.3.5 Motor Babbling with Shadow Hand Synergies

In this experiment we applied the motor babbling approach to the Shadow Hand equipped with force/torque sensors in the finger tips (Figure 10). The parameters to learn were anatomical synergies related to thumb abduction and index/middle finger abduction. The wrist was fixed with respect to the object reference frame. In the beginning of each trial the hand was fully opened and it would close until touching the object and a reliable metric value could be measured. One hundred trials were performed. In the end, the expected value of the Gaussian Process modeling the quality metric was computed on a dense grid over the parameters under optimization. The results are shown in Figure 28. One can observe that the region at the

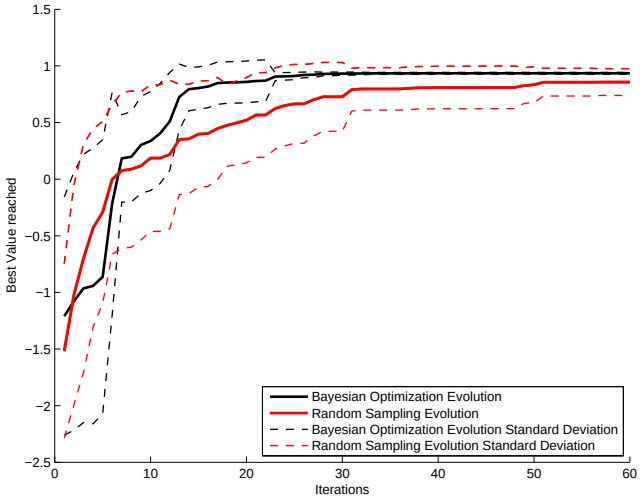

Figure 27: Comparing the evolution of best value found when sampling a mug with the Bayesian approach versus the random sampling method.

right of the synergy space, where the thumb was positioned in a more perpendicular orientation with respect to the object facet, have higher quality values in general. This map can now serve as a prior to predict the best grasps in similar objects.

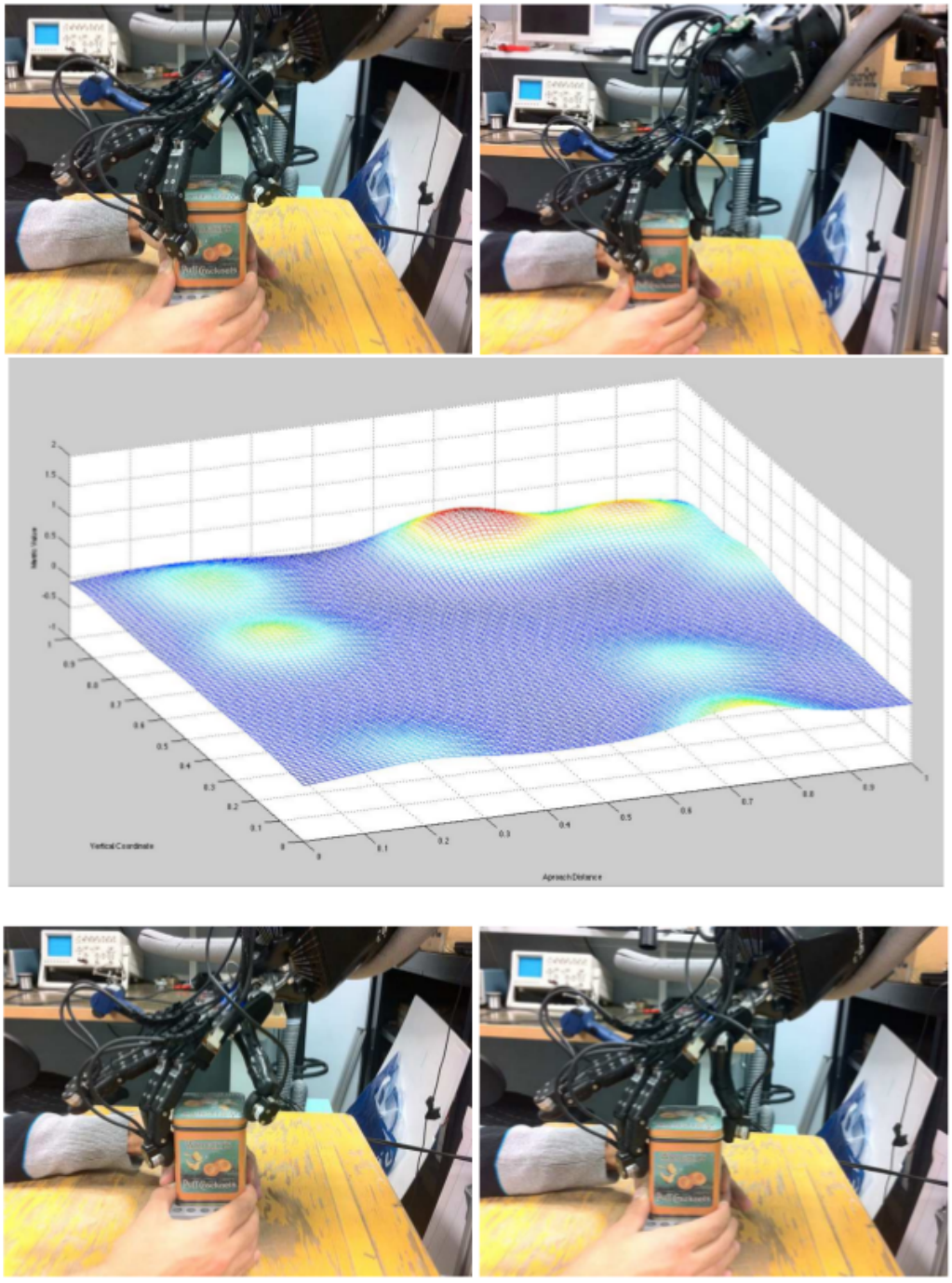

Figure 28: Through motor babbling in the hand-synergy space, the robot learns the best grasp configurations on the object. The stability metric values on the right half synergy space, corresponding to the pictures on the right, have higher quality in average, due to a more favorable thumb abduction.

5 Action Gist Guided Motor Babbling for In-hand Manipulation

We plan to use motor babbling for Shadow-hand in-hand manipulation learning, based on the action-gist [94] extracted from human demonstrations. Using slightly varying finger motions, the hand is trying to touch and move the object again and again, until the correct motion sequence corresponding to the human demonstration is obtained. Therefore, the fingers will be moved as scheduled, obeying the action gist, but varying in the amplitude of joint angle. So far PSO is selected for parameter exploration, and some experiments have been carried out. Even though the results are not satisfactory yet, there are many ways to improve the situation.

5.1 Introduction

Current robotic hands have become almost as flexible as human hands [114] [104] [129] [120]. This applies especially to the Shadow hand, where we can access both real and virtual hands to perform the hand movement supported by the robot operating system ROS. Therefore, in the field of in-hand manipulation it is possible to exactly mimic the hand behavior of humans.

We can find many sucessful cases in the object manipulation based on building a kinematic or dynamic model for a specific application [107] [118] [115] [131]. Another approach to move the object is to leave the modeling process to the robot itself. In other words, the robot can learn from human demonstrations.

To achieve this "learning from human demonstration" process, the adoption of the right kind of sensory input and the analyzing (modeling) of the recorded information are the key parts. The commonly applied sensors are cameras for hand and object posture [117], data-gloves for non-occlusive hand posture [103], tactile sensors for hand-object interaction [132], or ToF cameras for direct 3D information [101], and positional sensors for key part positioning [102]. Based on what have been perceived, we can start manipulation modeling. There are many kinds of modeling, such as contact point based [123] [126] [97], and moving trajactory based methods [130], [109]. We aim at both modeling the hand action and object state and making the model widely used on different-sized hands. We also believe there exists robust actions that can result in the object moving towards the target state, because humans have the ability to learn manipulation skills from others even if their hand sizes are slightly different. The current situation is that we have recorded human demonstrations, and we are going to apply the skill to a larger robotic hand. Therefore, we decide to generate a state-action model for a robotic hand, and then leave the robot to refine its own movement. In the progress of state-action modeling, we have developed the way to extract the *action gist* – the key hand movement consisting of several predefined basic motions [94]. When the robotic hand performs a

manipulation task, the action gist works as a guideline to instruct the fingers to move step by step. With this kind of input, the search space of the finger gaiting decreases. Nevertheless, in practical implementions the robot will have to convert the abstract movement into explicit joint values, which is where the learning comes into play. In each step, the robot fingers move as prescheduled, until the set of control parameters correctly work on the fingers to move the grasped object into the destinated state. The joint value exploration is the key purpose of this report. The Shadow hand has 24-DOF overall, of which 20-DOF are controllable due to the J1/J2 coupling of the fingers. For this research we only consider the in-hand gaiting movements of the fingers, ignoring the wrist motion, resulting in a 18-DOF state–space to specify the hand pose. However, for the action-state modeling, the dimension is even higher because we need to multiply the mechanical 18-DOF with the number of the actions in the specific application. In this case, we have to consider evolutionary algorithms to tolerate the dimensionality while approximating to the best solution. Here we employ the idea of *Particle Swarm Optimization* (PSO) [108] [127] to optimize the searching.

Compared with all other swarm intelligence algorithms, PSO is the closest candidate for finger joint value exploration. Firstly PSO covers the concept of particle speed and position, so we can directly combine the PSO parameters to the joint values; and after the evaluation of one loop of manipulation control using the current parameters, we can adapt the parameters by changing the speed. This process is similar to what the human hand can do. Secondly the amount of the particles promises this algorithm to be resilient to local minima. This point ensures that we can find a good solution after a sufficient number of iterations. Currently our purpose is to apply action gist to guide the finger movements, so it does not matter whether the selected learning algorithm is the best one. Later the PSO can be replaced by the Cuckoo search, which is claimed to outperform PSO according to a recent paper [96]. Even after PSO is selected, we are not sure the particle flight direction is approaching the correct solution.

We have applied action gist to limit the joint value variation. Meanwhile, the parameters needed to learn involve the joint value variation of each finger as well as the corresponding start time. We treat all of the joint value variations and start times together by translating the value into the proper form for the PSO model. Each babbling of the hand as the particle moves can reflect the corresponding cost of the PSO model; in this way the particle moves to the new babbling parameters until the final solution is found.

5.2 Related Work

Motor babbling originates from the concept named “body babbling” proposed by Meltzoff and Moore [116]. Under the claim that the imitation is a matching-to-target process, a loop integrated with infant actions, adult actions and propriocep-

tive feedback was introduced to match Meltzoff and Moore's active inter-modal mapping (AIM) hypothesis. As a key part of the infant action system, body babbling coordinates movements to the organ endstates so as to achieve the goal that the adult aims for. Because an infant needs the knowledge from adult, prior input is allowed to strengthen the exploration of body babbling. Therefore, the previous framework was extended by [128] [100], which approached to robot and considered infant motor acts as motor planning. Besides, it explicitly indicated that motor planning consists of three models: forward model (world dynamics), prior model (instructor's policy) and inverse model (action selection). These three components drive the robotic individual to take actions according to its own performance and the instructor's constraint. Actually, so far there are two working mechanisms in motor babbling application. The first kind of treatment is that motor babbling is a kind of robotic behavior only consisting of random movement [92] [113] [124] [121], the evaluation begins when all trials are done. Otherwise we will see that behavior improves with the iterations of babbling, like [90] [91] [119] position motor babbling as "Behavior Coordination". Currently we are aiming at in-hand manipulation learning, and we realize that exploring finger movement is necessary. As this searching process cannot promise achieving better solutions everytime, we consider the process as iterated motor babbling learning.

Before the robot hand starts motor babbling, we should prepare a schema for the robot so as to tell it what, when and how it can do according to the scenario, as in the cases of [110] [111] [105] [112]. In the process of motor babbling, the robot should exploit its own memory to reorganize prior knowledge to instruct itself [89]. Even if the robot begins without prior knowledge, it is possible to use forward and inverse models to generate its knowledge database [99].

Furthermore, we can refer to some cases related to manipulation babbling learning: [125] applied motor babbling in arm posture control combining with visual feedback. [122] applied Motor Babbling in 12-DOF robotic hand grasping. It is based on Hebbian theory [106], successful grasp parameters are strengthened, while the ones that tend to fail to do so are weakened. Furthermore, the hand control is through forward kinematics and inverse kinematics. [93] simulates a babbling reaching and grasping scenario with obstacles, the control parameters are iteratedly updated by a biological constraint – Hebb learning rules. [95] applied motor babbling in CPG-driven hand grasping simulation. Through adjusting the parameters of different CPG models, different-sized hands can learn how to rotate different-sized objects.

Here we are targeting the 5 finger in-hand manipulation for the Shadow robot hand. Therefore, the degrees of freedom and the finger-gaiting complexity are much larger than before, so we are going to propose a new solution for this new challenge. Also we want to keep the learning time for the real robot as low as possible, because we do not want to damage the robot. [98] give us a good example by applying the PILCO (probabilistic inference for learning control) framework for learning itera-

tion. In their viewpoint, babbling is data inefficient in the real test because it requires too many iterations. As a result, the manipulation process is executed in simulation iteratively before practical test. Afterwards we can compare the simulation and the real test with the same parameters to improve the simulated and practical result.

5.3 Robot Hand Control

The basic unit of an articulated robot hand is the finger joint. Therefore, in the low level control, we usually reconfigure hand posture by sending joint angle information. We can use the word "frame" to present the moment of sending one or more joint values to move the finger. Considering the applications of in-hand manipulation, the joint variations are more complicated than the pure grasping movement. This means the method of configuring finger joint angle frame by frame is not realistic. Currently our framework instructed by action gist is not on contact point scheduling, so we will not use inverse kinematics to planning the hand posture. Instead, because the action gist provides us with the sorted meta motions, and each meta motion limits the range of the joint variation, we can tell the robot to perform the manipulating movement in the order instructed by the meta motions. Supposing that the initial hand and object posture are ready, given the start time, end time, and angle variation corresponding to the finger, it is possible to see the exact hand movement as commands sending by frame. Here, the initial start time, end time and angle variation can be easily found from the data-glove value sequence along with the action gist extraction. Besides, we notice that for each finger, the end time of a meta motion is the start time of the successive meta motion, so the end time of each meta motion is redundant. Because joint values are not calibrated and the sizes of the demonstrator's and robotic hands are different, we do not expect that the initial data will work as soon as it is applied to the target application. However, the order of the meta motion is trustable, we believe that after iterations of parameter adjustment, we can find the correct values to control the robotic hand to complete the manipulation task. The details of the process will be disscussed below in subsection 5.4.1.

5.4 Action gist based Motor Babbling Learning

Based on action gist, we have already had a rough view of when and how the fingers should move. Then we combine the information with PSO exploration to refine the control parameters in the simulation. Fig.29 illustrates the framework of our work. Using the Gazebo simulator with physical engine, we have a testbed to babble the hand movement repeatedly without damaging the real robotic hand. Firstly we should initialize the hand and object, assuming that the object is already in hand. Then the hand is controlled by the command frames, each consisting of all joint variations to complete a trial. In order to avoid useless workload, there

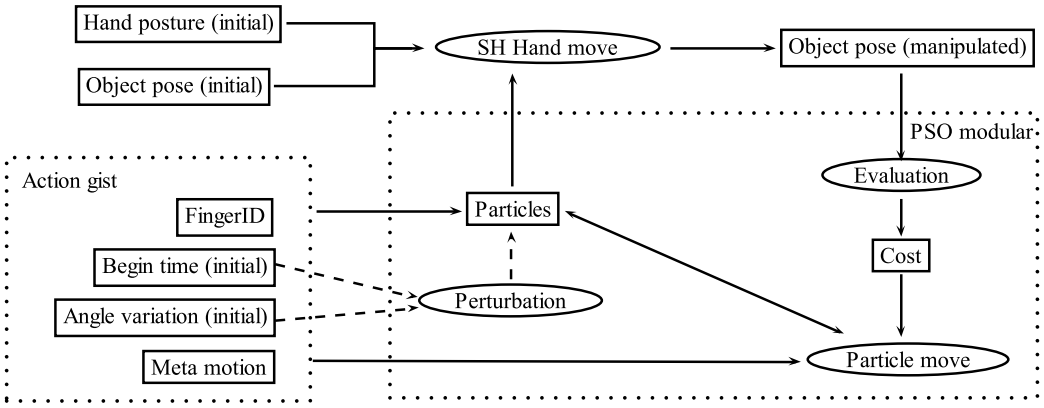

Figure 29: Main workflow of action gist based babbling learning in simulation. Generally, the meta motion sequence (action gist) consisting of which finger it is, when it moves, how the angle variation of corresponding joints they are, and which meta motion it belongs, are considered. The values from the entire meta motion form a particle, specifically, the begin time and joint angle variation will be slightly changed because there are multiple particles for the algorithm. Then all particles (control parameters) are put into the shadow hand simulation, the object is manipulated. For the result we can estimate, and give feedback to PSO modular so as to adjust the particle values. Because action gist constrain the range of each parameter in the particle, the exploration effective is improved. Finally we will have a acceptable result with iterations.

are comparisons between several frames to check whether the object is moved as scheduled. Once a state is far from our expectation, the hand posture and the object pose will be reset for another trial. In order to receive a good learning result, we pay attention to several key issues as detailed in the following subsections.

5.4.1 Joint angle control parameters

As mentioned in Sec.5.3, we apply a set of parameters in relatively compact format indicating joint angle variation, start time, and the corresponding finger. The robotic hand concentrates on learning these parameters by the feedback of executing them. Meanwhile we notice that we should design the parameters in a proper form for the robotic hand, because the prior knowledge is extracted from a data-glove. Additionally, we can compute the scale of the parameters.

5.4.2 Joint mapping from the data-glove to the robot hand

After human demonstration, action gist is extracted from the data-glove data. In order to pass this information to the robotic hand, we need to map the joint variation

from the data-glove to the corresponding joints on the robot hand. There are several reasons to do so. Firstly, instead of randomly initializing the control parameters, we configure them similarly to the raw values from the data-glove, which are not calibrated initially. Calibrated data-glove values are not strictly necessary in our case, because action gist itself is only based on the joint angle variation. However, if the map values are already close to the final solution, this of course provides a better initialization for the learning process. As shown in figure 30, the Cyberglove measures the abductions between finger pairs, while the abduction angles can be controlled for each finger individually on the Shadow hand. Several other joint angles of the Shadow hand have no direct correspondence with the Cyberglove sensors neither. Therefore, according to our experience in generating the action gist, we propose mapping their relations in following way.

1. Keep all joints in exactly the same positions on the hand.

2. Abductional joints are only used to assign the initial value to the robotic hand, thumb-index to the robotic thumb finger, index-middle to the robotic index finger, index-middle to middle, ring-middle to ring, and little-ring to little finger.

3. Thumb-index abduction controls both Shadow hand thumb abduction joints.

4. The carpometacarpal joint of the little finger is always set to 0.

5.4.3 The dimenson of the control parameters

Since we use the start time, joint angle variation, and the corresponding finger to instruct the in-hand movement, for a specific application, we can calculate how many parameters are necessary. First, we assume the order of meta motions is correct after action gist extraction, so we have a fixed order of the parameters. Second, depending on the finger, we can determine the number of joints as shown in figure 31. Therefore, presuming that we have a meta motion sequence $\mathbf{m}$ consisting of L_m motions, and that each motion is paired with a start time, we can simply get the total dimension as follows:

$$d_{\mathbf{m}} = \sum_{i} \left(1 + L_{joint}\left(\tau_{finger}\left(m_i\right)\right)\right) \tag{45}$$

here $\tau_{finger}\left(\cdot\right)$ indicates the finger of meta motion m_i, and we can know joint number $L_{joint}(\cdot)$ from Fig. 31. Now we establish a $d_{\mathbf{m}}$-sized vector to match the latter learning algorithm.

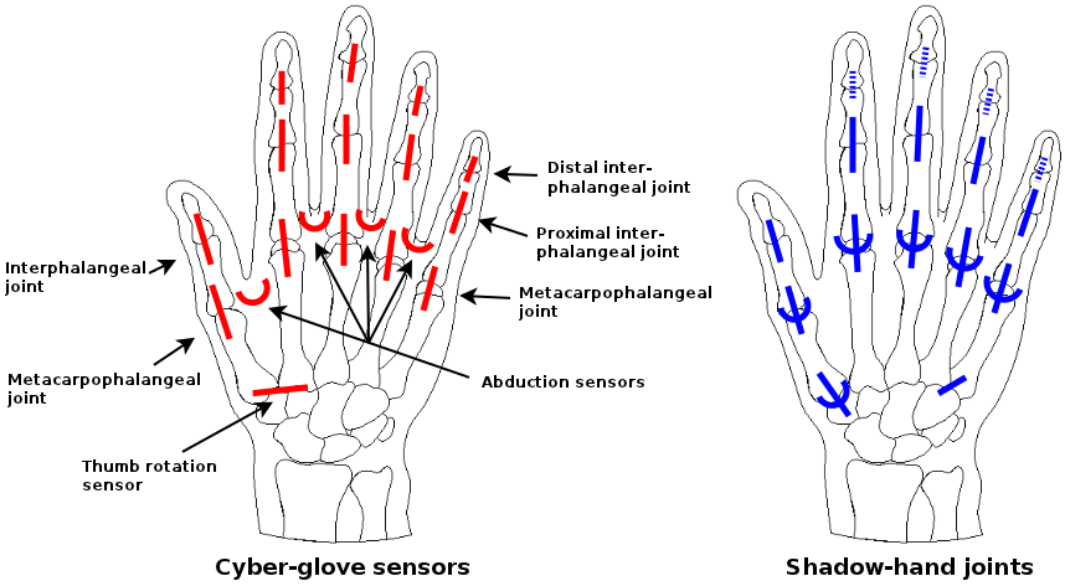

Figure 30: The sensor and joint layouts of the Cyberglove and the Shadow C5/C6 hand. As the red tags indicate, the Cyberglove sensors cover all visible joints of the human hand and also provide the thumb rotation measure. By comparison, we can easily find the difference in the joint structure of the Shadow hand, e.g. each finger has its own abductional degree of freedom, the thumb has 5 DOF and the little finger has a Carpometacarpal joint. Finally, we do not need to assign the distal joint angles to the Shadow hand, as they are underactuated and will be automatically be moved, controlled the proximal interphalangeal joints.

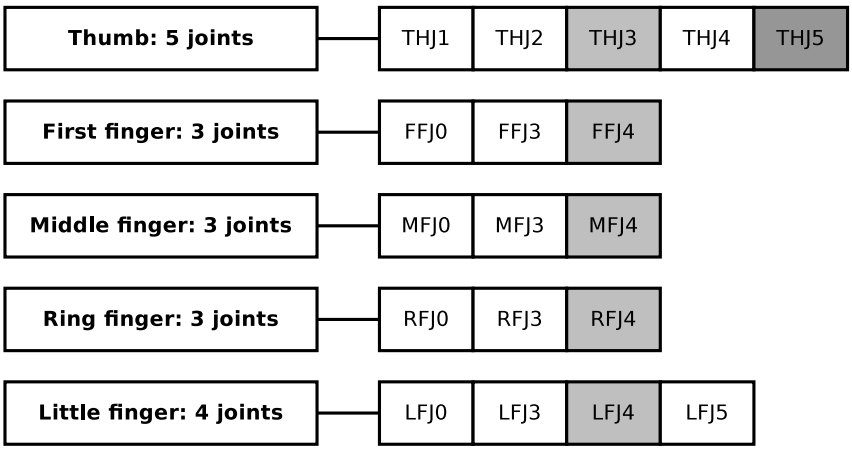

Figure 31: The joint map for the corresponding fingers of the Shadow hand. The distal and middle links (J1 and J2) from the first to the little finger are coupled, a single value J0 controls both J1 and J2. The abduction joints are colored as grey, and particularly THJ5 is a rotational joint.

5.4.4 Translation between control parameters and shadow hand joint angle frames

Action gist abstracts the continuous hand movement into meta motions, and we are going to replay similar movements as learned from human demonstration. To create a set of motion-related control parameters is good for learning, but we have to convert it back into the joint angle frames to communicate with the robotic hand.

The control parameters consist of start time and joint angle variation, and the initial joint angles of the robot hand at the moment. Therefore the general control flow is presented as Alg. 1. In the procedure we can see several new variables and functions. Because the finger joint angle is sent to the robot hand frame by frame, first of all, we should normalize the time in order to know whether the working frame is controlled by the meta motion m_i (paired with c_i). Therefore we take $T_{nor}(j) = j/N_{frame}$ to normalize the frame time, $\tau_{start\,time}(\cdot)$ and $\tau_{end\,time}(\cdot)$ to obtain the normalized time of meta motion m_i. Second, we should tell the robotic hand to update the corresponding finger joint angles as $\mathbf{A}(\cdot)$. Considering that the joint variation relies the time and motion, there are many possibilities to design this function. Currently we only use the simplest form – a linear model to deal with the joint variation, i.e.

$$\mathbf{A}\left(c_i, m_i, T_{nor}(j)\right) = \frac{T_{nor}(j) - \tau_{start\,time}(c_i)}{\tau_{end\,time}(c_i) - \tau_{start\,time}(c_i)} \times \tau_{joint\,variation}(c_i) \qquad (46)$$

where $\tau_{joint\,variation}(c_i)$ involves all relevant joint angle variations of motion c_i.

Algorithm 1 General control flow as provided motion control parameters

Require: The control parameter sequence $\mathbf{c}$ corresponding to action gist $\mathbf{m}$
Require: The number of frames to communicate with the robotic hand N_{frame}
1: **for** $j := 1$ to N_{frame} **do**
2: **for** each $c_i \in \{\tau_{start\,time}(c_i) < T_{nor}(j) < \tau_{end\,time}(c_i)\}$ **do**
3: Update the finger joint angles as variation $\mathbf{A}\left(c_i, m_i, T_{nor}(j)\right)$
4: **end for**
5: **end for**

In addition, according to the analysis of raw data-glove values, we notice that the linear model is not enough to exactly describe the joint angle variation. This simplification is insufficient for some complicated applications, e.g. when a finger is required to move with a certain acceleration. Thus, we will apply higher order or other fitting models to abstract the joint angle variation. Just for the sake of decreasing the dimension of parameters, we aim at solving simpler cases first, by assuming that each finger moves at a constant speed in a motion slot. In the future we will extend this framework to fit more applications.

5.5 PSO model for babbling learning

Like other evolutionary algorithms, PSO has the concept of a group involving many candidates to approach the goal solution. However, instead of eliminating weak and then generating new candidates, all candidates update themselves by moving to new solutions as referring to the current best solution (or the center of a best solution cluster). It is unnecessary to repeat the original procedure of PSO in this document, so we concern ourselves with how to make PSO work effectively in our application. A slightly different version from the standard PSO is proposed as Alg.2.

5.5.1 Action gist limits the exploration space

We mark this point in Alg.2 as ***1****. Without action gist, the search space for each dimension would be $(-\theta_d, \theta_d)$. Because meta motion indicates the direction of the corresponding finger, the bi-directional search space is limited to one direction. Thus now the search space for each dimension is $(0, \theta_d)$. Considering the dimension of the joint angle control parameters, we can say action gist help us a lot to decrease the exploration space. However, as the exploration space is still very large, we need more techniques to serve our application.

5.5.2 Incremental parameter adjustment for PSO exploration

In the previous section 5.4.1 we discussed the input for PSO. The parameters in this format cover all motions across the entire manipulation process. However, a trial for replaying the parameters in simulation takes easily more than one minute on typical computers, because we apply a dynamic environmental simulation with all parameters very close to the real world. Meanwhile, we use "states" to check the correctness of the motion execution. Once the state differs from the plan, we can stop this trial to save time. In this case, we notice that only the control parameters applied from the start state to the checking state have influenced the robotic hand action. For the remaining control parameters, they did not have a chance to prove themselves. Therefore, even though the original PSO will give a speed to every parameter in order to make them converge to a better solution, we should not modify any unproved parameters. We mark this point in Alg.2 as ***2****, where function $floor(r)$ returns the nearest integer just below the real number r. The value of $f(p_i)$ will be explained in the next section 5.5.3. Because the value provides us with the information of how many parameters are involved, we can avoid updating the unused parameters. Only when the current parameters complete the state test and proceed to the next state test, the successive parameters are activated to join the learning iteration. Therefore, the learning progress is considered as an incremental process.

Algorithm 2 Incremental PSO for in-hand motor babbling learning

Require: Maximum of iteration N_{Iter}
Require: Particle number $N_{particle}$
Require: Particle dimension $d_{\mathbf{m}}$
Require: Joint variation limitation A_{lim} ▷ ***1****

for each particle $i = 1, \ldots, N_{particle}$ **do**
    Initialize the particle's position with a **uniformly distributed** random vector $x_i \sim U(b_{lo}, b_{up})$ ▷ $b_{lo} \leftarrow 0$
        ▷ $b_{up} \leftarrow 1$ when x_i is a start time parameter, in other cases b_{up} has a corresponding value from A_{lim}
    Initialize the particle's best known position to its initial position: $p_i \leftarrow x_i$
    Simulate a trial as x_i to get $f(p_i)$ ▷ ***3****
    if $f(p_i) < f(g)$ **then**
        update the swarm's best known position $g \leftarrow p_i$
    end if
    Initialize the particle's velocity: $v_i \sim U(-|b_{up} - b_{lo}|, |b_{up} - b_{lo}|)$
end for
$counter \leftarrow 1$
repeat
    $counter \leftarrow counter + 1$
    for each particle $i = 1, \ldots, N_{particle}$ **do**
        for each dimension $d = 1, \ldots, floor(f(p_i))$ **do** ▷ ***2****
            Pick random numbers: $r_p, r_c \sim U(0, 1)$
            Update the particle's velocity: $v_{i,d} \leftarrow \omega v_{i,d} + \varphi_p r_p(p_{i,d} - x_{i,d}) + \varphi_c r_c(c_d - x_{i,d})$ ▷ c is the center of a best solution cluster
        end for
        Update the particle's position: $x_i \leftarrow x_i + v_i$
        Simulate a trial as x_i to get new $f'(p_i)$ ▷ ***3****
        if $f'(p_i) < f(p_i)$ **then**
            Update the particle's best known position: $p_i \leftarrow x_i$
            if then$f'(p_i) < f(g)$
                Update the swarm's best known position: $g \leftarrow p_i$
            end if
            Reorganize the best solution cluster c
        end if
    end for
until $count >= N_{Iter}$ or $f(g) = max$
Output the best found solution g

5.5.3 Evaluation function

The evaluation function gives an overall estimation of the learning success after a trial of simulation. Here we intend to design the checking rules as simple as possible, for the following reasons:

1. In-hand manipulation is a dynamic process. Driven by the finger forces, the manipulated object keeps its state only for a certain time. The (simulated) world keeps running, and too complex state calculations may take longer than the time available for reacting.

2. Real robotic tests use different methods to perceive states. In the simulation, we can employ several software functions to acquire the state, e.g. object position, contact area. However, in practice we have to use the available sensors like cameras and tactile sensors to reach our goal. As we know, sensory error is an issue in real world data processing.

So far what we have applied is object position information, object rotation information, and finger tip contact information. For some cases, we just consider the object position, and assume that the object is correctly moved if the object does not fall down until after the manipulation movement.

Therefore, we have two evaluation functions. The first one only has one check point at the end of the trial. In this case we just make the evaluation function return how long the object is kept in hand. Another evaluation function works for Alg.2 as ***3****, it is an incremental function with respect to the number of activated parameters. Before the description of this function, we should declare the mechanism of the score accumulation.

We divide the entire in-hand manipulation process into several parts. At the end of each part we use several criteria (states) to check whether the hand action is correct. As long as the action fully passes the checking of one state, we can activate the successive conrol paramters for the next state. According to Sec.5.5.2, we will give the basic score:

$$S_b = \text{the number of activated joint angle control parameters} \tag{47}$$

Even if the hand motions cannot pass the state checking, we can still estimate how well the current motion has achieved the goals. During the motion execution, we take samples of the properties we are interested in, e.g. object position, which finger tips touch the object, etc. Supposing that the sample record is $\mathbf{s}$, while the benchmark for checking is $\mathbf{t}$. We can compare these values in a same scale L, so as to obtain the achievement score:

$$S_a = \sum_{i=1}^{L} q_i / L, \quad q_i = \begin{cases} 1 & \text{if } s_i = t_i \\ 0 & \text{if } s_i \neq t_i \end{cases} \tag{48}$$

Now that we have Eq.47 and Eq.48, we have an overall evaluation function for learning feedback:

$$f(p_i) = S_a + S_b \tag{49}$$

5.5.4 PSO parameters

Through Alg.2, we can see that the PSO parameters which can be tuned are: the iteration number N_{Iter}, particle number $N_{particle}$, velocity control ω, φ_p and φ_c. Because the dimension of the particle is a variable depending on the complexity of the manipulation, $N_{particle}$ will go along with the dimension of control parameter variation. According to our test, the convergence speed is very fast but we cannot have the best solution. Therefore, rather than increasing N_{Iter}, we prefer to restart the babbling learning process by tuning $N_{particle}$. For the three velocity parameters, we always keep them as $\omega = 0.4$, $\varphi_p = 0.4$ and $\varphi_c = 0.4$.

5.6 Experiment

In this section, we present a complete babbling learning simulation using the ROS Shadow stack and the Gazebo simulator. As mentioned in the previous sections, for each iteration, the first step is to initialize the hand in a stable state that grasps and holds the target object by fingers. To achieve this, we set the object in a position, then configure the hand joint angle values to have the object grasped by the fingers. Then the robot starts to try the motion sequence by itself, and we can see the object being manipulated.

5.7 Cylinder rotation

Since action gist contains the patterns reflecting joint angle variation, but without relating to the manipulated object, we are going to re-use the demonstration of screw-cap rotation (as Fig.32) to instruct the rotation of a cylinder. We can find that the thumb, first, middle and ring finger are used in this application; the four fingers tightly grip the screwcap and move. Therefore, when the knowledge is translated into a cylinder rotation, we should also see that four fingers being used and the cylinder being rotated in the same direction as the screw-cap in the human demonstration.

Figure 32: Cylinder rotation learning from screw-cap rotation. A demonstrator wearing a data-glove rotates the screw-cap of a bottle. The thumb, first, middle and ring finger are involved in this application.

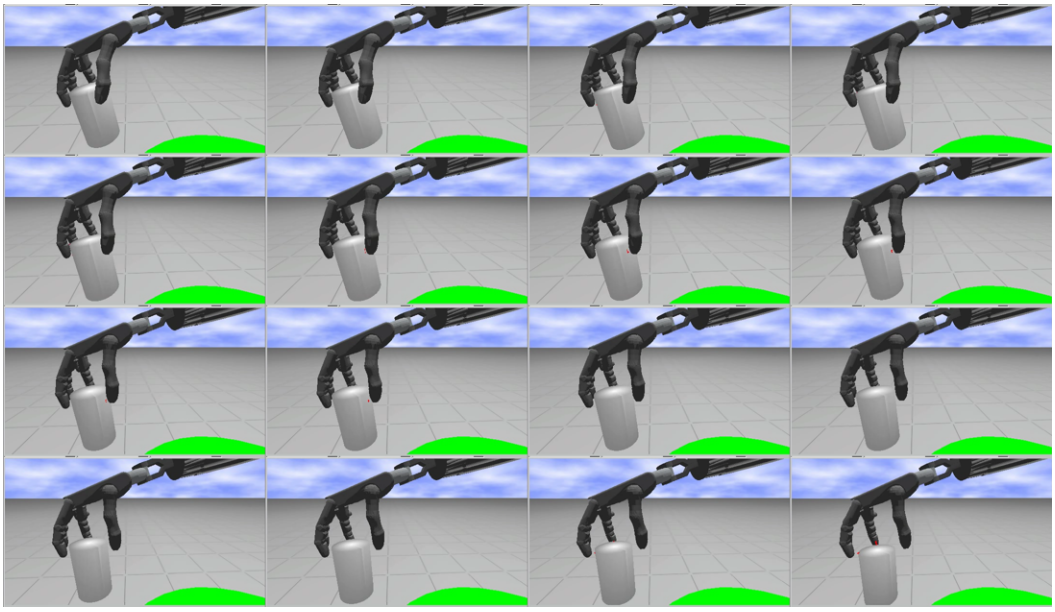

Figure 33: Cylinder rotation in Gazebo simulation. From the grey edge, it can be seen that the cylinder is rotated. However, it falls down before all motions end.

We can offer an interim result in the progress of parameter training as shown in figure 33. In the images we can see the finger motion reconfiguring the object as planned. However, after many iterations, the final solution is somehow an ill action. Even though the hand motions can keep the object in hand longer, and we can see the object being rotated, we do not accept the final solution to be as good as the interim result we show here. As shown by the learning curve in figure 34, we can see that PSO does work for parameter improvement. However, we need to define more states to achieve the ideal result.

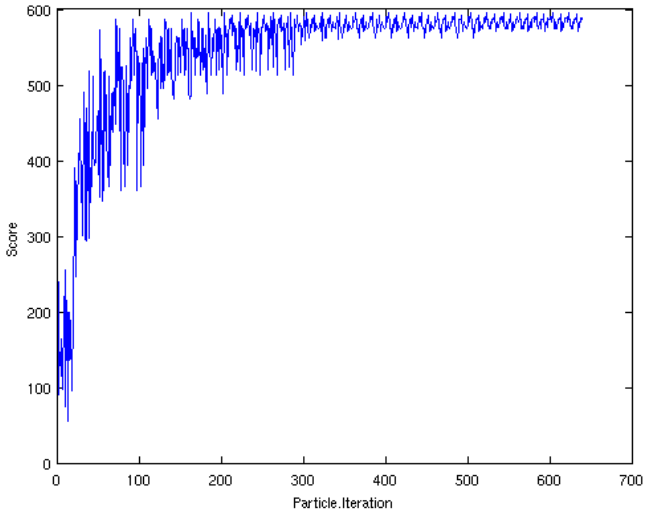

Figure 34: The babbling learning curve corresponding to cylinder rotation. This figure applies the old cost function, i.e. evaluation after all motions are executed. The cost function measures how long the action can last (before the object falls down), as well as whether the cylinder is rotated. The x axis indicates the number of iterations. By the curve shape we find that the particles converge very fast.

5.8 Conclusion

We proposed an in-hand manipulation babbling learning framework consisting of a priori knowledge (action gist) and self-learning mechanism (particle swarm optimization). With the help of action gist, we decrease the angle joint control parameters as a sequence that includes start time and corresponding finger joint angle variation. The compressed sequence is easy to convert back into joint angle frames, and then we can use simulation to execute and examine the performance of this set of parameters. Meanwhile a modified particle swarm optimization for our application is employed as the core of babbling learning, so as to refine the joint angle control parameters to achieve our goal of in-hand manipulation.

The test scenarios are not enough yet to prove the method we proposed. According to our experiment, we can see that PSO is improving the solution even though it is still not optimal. So far we only used a linear model to describe the joint angle variation, with 2 scenarios (cylinder rotation and star rotation), and only employed PSO. Therefore there is much room for extending our research.

6 Summary

The research approach of the HANDLE project towards grasping and in-hand manipulation with multi-fingered robot hands is characterized by two essential steps. First, human demonstrations of manipulation tasks are analysed to extract an initial set of basic robot skills. Second, the robot uses exploratory actions including motor babbling to gain additional knowledge about objects and their affordances, in order to improve its skills and performance. This is complemented by multi-modal sensing and low-level control strategies for precise motions and force-control of the fingers. The present report describes the final solutions developed within work-package WP3 of the project for *improving skills*, namely the use of postural synergies for grasping and towards manipulation motions, and the use of motor-babbling to improve the quality (e.g. stability) of generated grasps.

Classical grasp planning algorithms for the Shadow hand must consider the extrinsic 6-DOF for the relative pose of hand and object plus the 24-DOF required to specify all finger joint-positions, and therefore are facing all problems related to searching in high-dimensional spaces. As explained in a previous report [152], the extraction and parametrization of suitable *grasp-synergies* was selected as the most promising way to grasp and handle complex objects. To bypass the difficult issue of mapping from human demonstrations to the different kinematics of the robot hand, a large set of grasps on objects of different sizes and shapes was recorded using tele-operation, where the human operator would adjust the hand pose until it was accepted as a "human-like" grasp. This anlysis was performed for eight dexterous fingertip grasp-types, including the pinch and tripod grasps most suitable for in-hand object reorientation.

The synergy matrices were then derived from PCA on these datasets. The material presented in section 3 builds on the analysis from D24 [152] and summarizes the use of the postural synergies for grasp-planning and simple manipulation motions. In particular, the data analysis presented in subsection 3.2.1 indicates that the hand workspace is surprisingly narrow in the x-direction, while grasp positions correlate well with object size (and shape) in the y- and z-direction. This result together with the matching approach vectors for the eight different grasp types provides all information required for the initial pre-shape and reach-to-grasp motions. To cope with calibration errors and noisy object recognition, a reactive grasp approach using tactile-sensing to avoid hand-object collisions was proposed in subsection 3.3. Using the linear synergy matrices and existing FK/IK solvers, switching between joint-space, cartesian-space, and synergy-space is straightforward, and the most suitable representation can be used at any moment. Several experiments indicate that the postural synergies can indeed be used to parametrize and execute manipulation motions on the Shadow hand.

The use of motor babbling in continuous action spaces is the topic of section 4, targeting the *optimization of grasp-stability* on unknown objects. Due to uncertainty

on both the robot kinematics and motor control, but even more on object perception, it is very hard to analytically compute good grasps and to execute them successfully. After a short review of the underlying contact models and grasp-wrench space, the Bimodal Wrench Space metric was chosen as the most suitable quality measure. The concept of Bayesian optimization and Gaussian Process regression was then applied to the grasp optimization problem, and it was argued that the algorithm was still feasible despite the high-dimensionalty of the search space implied by the Shadow hand, given that uniform sampling of the search space is replaced by a smart exploration technique. In fact, the approach may be compared to human learning stages where infants learn by trial and error to find what the best grasping strategies are. Initial grasps are often unsuccessful, but after a few trials the system learns to adapt to the uncertainties in the environment. Experimental results from simulation and the real hand has been presented in section 4.3 for the Barrett-hand and the Shadow-hand and a set of prototypical grasp objects. Initial results demonstrate that the algorithm can also be applied to improve grasping based on the postural synergies for the Shadow hand.

Finally, section 5 proposed an exploration algorithm to generate in-hand manipulation motions for the Shadow hand based on the *action-gist concept*. The idea is to extract key motions from human demonstrations recorded via a dataglove and optionally also additional sensors, where the traces are first classified locally according to the finger-flexure and finger-abduction. Based on those key motions, a Particle Swarm Optimization learning algorithm was used to explore the hand joint-space and to reconstruct suitable finger-motions in order to find and maintain contact points with the target objects. First tests from the Gazebo simulator were presented for screw-cap rotation motions involving finger-gaiting.

Future work

The algorithms and experiments presented in this report demonstrate that many grasping and manipulation tasks can be performed autonomously with the Shadow hand, where motor-babbling has proven to become an important mechanism to improve the quality of initial grasps. Despite the obvious limitations of the hardware and actuators, the main problem encountered during the experiments regards the lack of integrated sensing; the fingertip force/torque sensors developed for the HANDLE demonstrator hand work well, but many human grasps require sensors spread over the whole hand. A better integration of visual information and suitable algorithms for tracking object-pose based on fused vision, proprioception, and tactile information will be required for the next major improvement towards robot in-hand manipulation. The development of a fully-equipped *sensing hand* seems to be the necessary step before multi-fingered robot hands can match human performance.

Acknowledgements

To the whole HANDLE consortium for the motivating and fruitful discussions.

References

[1] C. L. Taylor and R. J. Schwarz. The anatomy and mechanics of the human hand. *Artificial limbs*, 2(2):22–35, 1955.

[2] C. Ferrari and J. Canny; *Planning Optimal Grasps* In Proceedings of the IEEE Int. Conference on Robotics and Automation, pages 2290–2295, Nice, France, 1992.

[3] K.B. Shimoga, *Robot grasp synthesis algorithms: a survey*, International Journal of Robotics Research, vol.15, 230–266, 1996

[4] A. Bicchi, V. Kumar, *Robotic grasping and contact: A review*, IEEE International Conference of Robotics and Automation, 348–353, 2000

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

[6] S. Arimoto, *Control Theory of Multi-fingered Hands*, Springer 2008

[7] T. Iberall, *Human prehension and dexterous robot hands*, International Journal of Robotics Research, vol. 16, 285-–299, 1997.

[8] S.C. Jacobsen, J.E. Wood, D.F. Knutti, and K.B. Biggers, *The UTAH/M.I.T. Dextrous Hand: Work in Progress* The International Journal of Robotics Research, Vol.3, 21–50, 1984

[9] M. Kondo, J. Ueda, T. Ogasawara, *Recognition of in-hand manipulation using contact state transition for multifingered robot hand control*, Robotics and Autonomous Systems 56, 66-81, 2008

[10] M. Hüser, T. Baier, J. Zhang; *Learning of demonstrated Grasping Skills by stereoscopic tracking of human hand configuration*, IEEE Intl. Conference on Robotics and Automation, 2795-2800, 2006

[11] H. Kjellström, J. Romero, D. Kragic, *Visual Recognition of Grasps for Human-to-Robot Mapping*, Proc. 2008 IEEE/RSJ International Conference on Intelligent Robots and Systems, 3192–3197, 2008

[12] J. Aleotti, S. Caselli, *Grasp Recognition in Virtual Reality for Robot Pregrasp Planning by Demonstration*, IEEE Intl. Conference on Robotics and Automation, 2801-2806, 2006

[13] R.S. Dahiya, G. Metta, M. Valle, G. Sandini, *Tactile Sensing—From Humans to Humanoids*, IEEE Transactions on Robotics, vol. 26, no.1, 1–20, 2010

[14] K. Matsuo, K. Murakami, T. Hasegawa, K. Tahara, R. Kurazume, *Segmentation method of human manipulation task based on measurement of force imposed by a human hand on a grasped object*, Proc. 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems, 1767–1772, 2009

[15] Shadow Robot Dextrous Hand, www.shadowrobot.com

[16] John Lloyd and Vincent Hayward, *Multi-RCCL User's Guide*, McGill Research Centre for Intelligent Machines, McGill University, Montreal, Canada, April 1992

[17] F. Röthling, *Real Robot Hand Grasping using Simulation-Based Optimisation of Portable Strategies*, Ph.D Thesis, Technische Fakultät, Universität Bielefeld, 2007

[18] D. Bartenieff and I. Lewis, *Body Movement: Coping with the Environment*, Gordon and Breach Science, New York, 1980

[19] N. Hendrich, D. Klimentjew and J. Zhang, *Multi-sensor segmentation of human manipulation tasks*, Proc. IEEE MFI-2012, Salt-Lake City, 2012

[20] M. Santello, M. Flanders, J. F. Soechting, *Postural hand synergies for tool use*, Journal of Neuroscience, vol. 18 no. 23, pp. 10 105–10 115, 1998.

[21] J.M. Elliott and K.J. Connolly, *A classification of manipulative hand movements*, Developmental Medicine & Child Neurology, 26: 283-296, 1984.

[22] M. Ciocarlie, C. Goldfeder, P.K. Allen, *Dimensionality reduction for hand-independent dexterous robotic grasping*, Proc. 2007 IEEE/RSJ International Conference on Intelligent Robots and Systems, 3270–3275, 2007

[23] M. Ciocarlie and P.K. Allen, *Hand Posture Subspaces for Dexterous Robotic Grasping*, International Journal of Robotics Research, vol. 28, 851–866, 2009

[24] Andrew T. Miller; *GraspIt!: A Versatile Simulator for Robotic Grasping*. Ph.D. Thesis, Department of Computer Science, Columbia University, June 2001.

[25] C. Goldfeder, M. Ciocarlie, H. Dang, P.K. Allen, *The Columbia Grasp Databse*, IEEE Internatioonal Conference on Robotics and Automation, 1710–1716, 2009

[26] F.Röthling, R.Haschke, J.J.Steil, and H.J.Ritter, *Platform Portable Anthropomorphic Grasping with the Bielefeld 20-DOF Shadow and 9-DOF TUM Hand*, Proc. IROS-2007, 2951-2956, San Diego, 2007

[27] M. Ciocarlie, H. Dang, J. Lukos, M. Santello, P. Allen, *Functional Analysis of Finger Contact Locations during Grasping*, Prod. Eurohaptics Conference and Symposium on Haptic Interface for Virtual Environment and Teleoperator Systems, 401–405, 2009

[28] Ch. Borst, M. Fischer and G. Hirzinger; *Calculating Hand Configurations for Precision and Pinch Grasps*. Proceedings of the 2002 IEEE/RSJ International Conference on Intelligent Robots and Systems, Lausanne, Switzerland, 2002.

[29] Ch. Borst, M. Fischer and G. Hirzinger; *Grasp Planning: How to Choose a Suitable Task Wrench Space*. Proceedings of the IEEE Intl. Conference on Robotics and Automation (ICRA), New Orleans, USA, 2004.

[30] M. Schöpfer, H. Ritter, G. Heidemann, *Acquisition and Application of a Tactile Database*, IEEE International Conference on Robotics and Automation, 1517–1522, 2007

[31] D.D. Nguyen, T.C. Phan, J.W. Jeon, *Fingertip Detection with Morphology and Geometric Calculation*, Proc. 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems, 1460–1465, 2009

[32] Y. Sun, J.M. Hollerbach, S.A. Mascaro, *Estimation of Fingertip Force Direction With Computer Vision*, IEEE Transactions on Robotics, vo. 25, no.6, 1356–1369, 2009

[33] N. S. Pollard; *Parallel Algorithms for Synthesis of Whole-Hand Grasps*. Proceedings of the IEEE International Conference on Robotics and Automation, Albuquerque, NM, 1997.

[34] Y. Lui and M. Lam; *Searching 3-D Form Closure Grasps in Discrete Domain*. In Proceedings IEEE International Conference on Intelligent Robots and System, Las Vegas, Nevada, October 2003.

[35] L. Han, J. Trinkle, Z. X. Li; *Grasp Analysis as Linear matrix Inequality Problems*. IEEE Transactions on Robotics and Automation, vol. 16, no. 6, pp663-674, 2000.

[36] R. Haschke, J. Steil, I. Steuwer, H. Ritter; *Task-Oriented Quality Measures for Dextrous Grasping*, Proc. IEEE Conference on Computational Intelligence in Robotics and Automation, 2005.

[37] A.T. Miller, S. Knoop, H.I. Christensen, P.K. Allen, *Automatic grasp planning using shape primitives*, IEEE Internation Conference on Robotics and Automation, 1824–1829, 2003

[38] J. Zhang, B. Rössler; *Self-valuing learning and generalization with application in visually guided grasping of complex objects*. Journal of Robotics and Autonomous Systems, 47: 117–127, 2004.

[39] J. Kim, J. Park, Y. Hwang, M. Lee; *Advanced Grasp Planning for Handover Operation Between Human and Robot: Three Handover Methods in Esteem Etiquettes Using Dual Arms and Hands of Home Service Robot*, 2nd Intl. Conference on Autonomous Robots and Agents, 2004

[40] T. Baier, J. Zhang, *Resuability-based Semantics for Grasp Evaluation in Context of Service Robotics*, IEEE International Conference on Robotics and Biomimetics, 2006

[41] T. Baier, J. Zhang, *Learning to Grasp Everyday Objects using Reinforcement-Learning with Automatic Value Cut-Off*, IEEE/RSJ International Conference on Intelligent Robots and Systems, 2007

[42] D. R. Faria and J. Dias, *3D Hand Trajectory Segmentation by Curvatures and Hand Orientation for Classification through a Probabilitistic Approach*, Proc. IEEE/RSJ IROS 2009, 1284-1289

[43] A. Gams and A. Ude. Generalization of example movements with dynamic systems. In *IEEE-RAS International Conference on Humanoid Robots*, pages 28–33, 2009.

[44] M.J. Gielniak, C.K. Liu, and A.L. Thomaz. Stylized motion generalization through adaptation of velocity profiles. In *IEEE International Symposium on Robots and Human Interactive Communications*, pages 304–309, sept. 2010.

[45] Elena Gribovskaya, S. M. Khansari-Zadeh, and Aude Billard. Learning nonlinear multivariate dynamics of motion in robotic manipulators. *International Journal of Robotics Research*, 30(1):80–117, 2010.

[46] Auke Jan Ijspeert, Jun Nakanishi, and Stefan Schaal. Movement imitation with nonlinear dynamical systems in humanoid robots. In *IEEE International Conference on Robotics and Automation*, pages 1398–1403, 2002.

[47] Christopher D. Mah and Ferdinando A. Mussa-Ivaldi. Generalization of object manipulation skills learned without limb motion. *The Journal of Neuroscience*, 23(12):4821–4825, 2003.

[48] Woojin Park, Don B. Chaffin, Bernard J. Martin, and Julian J. Faraway. A computer algorithm for representing spatial-temporal structure of human motion and a motion generalization method. *Journal of Biomechanics*, 38(11):2321–2329, 2005.

[49] Peter Pastor, Heiko Hoffmann, Tamim Asfour, and Stefan Schaal. Learning and generalization of motor skills by learning from demonstration. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 1293–1298, 2009.

[50] Xingda Qu and M.A. Nussbaum. Simulating human lifting motions using fuzzy-logic control. *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, 39(1):109–118, 2009bitemRichardA1975225

[51] Richard A. Schmidt. A schema theory of discrete motor skill learning. *Psychological Review*, 82(4):225–260, 1975.

[52] S. Schaal, A. Ijspeert, and A. Billard. Computational approaches to motor learning by imitation. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 358(1431):537–547, 2003.

[53] R.A. Schmidt and T.D. Lee. *Motor control and learning: a behavioral emphasis*. Human Kinetics, 1999.

[54] M. Riley and G. Cheng. Extracting and generalizing primitive actions from sparse demonstration. In *IEEE-RAS International Conference on Humanoid Robots*, pages 630–635, 2011.

[55] Videre Systems, www.videredesign.com/vision/sth_mdcs3.htm

[56] Polhemus Liberty Electromagnetic Motion Tracking System, www.polhemus.com/?page=Motion_Liberty

[57] Phasespace Impulse optical tracker, www.phasespace.com/productsMain.html

[58] Cyberglove systems, www.cyberglovesystems.com

[59] Tekscan Grip, www.tekscan.com/medical/system-grip.html

[60] Nintendo Corp., Wiimote, www.nintendo.com/wii/what/controllers

[61] HANDLE project, *D3 — Augmented sensing object*, www.handleproject.eu, 2009

[62] Andrew T. Miller, *Graspit!: A versatile simulator for robotic grasping*, IEEE Robotics and Automation Magazine, vol. 11, 110–122, 2004

[63] N. Koenig, and A. Howard, *Design and use paradigms for gazebo, an open-source multi-robot simulator*, Proc. IROS 2004, 2149–2154

[64] Bullet Physics Library, http://bulletphysics.org/, 2006

[65] JBullet physics engine, http://jbullet.advel.cz, 2008

[66] Hanno Scharfe, *Physikbasierter Simulator für Greif- und Manipulationsverfahren mit Mehrfinger-Roboterhänden*, Diploma thesis, University of Hamburg, 2010

[67] H. Scharfe, N. Hendrich, and J. Zhang, *Hybrid physics simulation of multi-fingered hands for dexterous in-hand manipulation*, Proc. ICRA 2012, to appear

[68] Lorenzo Sciuto, *Robotic Hand and Sensorized Glove: A Calibration for Managing Robotic Grasp in Teleoperation*, MSc. thesis, University of Siena, 2011

[69] Eugen Richter, *Hand pose reconstruction using a three-camera stereo vision system*, diploma-thesis, University of Hamburg, 2011

[70] Hao Dang, Jonathan Weisz, and Peter K. Allen, *Blind Grasping: Stable Robotic Grasping Using Tactile Feedback and Hand Kinematics*, Proc. ICRA-2011, 2011

[71] C. Ott, *Cartesian Impedance Control of Redundant and Flexible-Joint Robots*, Springer Tracts in Advanced Robotics 49, Springer 2008

[72] Eric Brochu, VM Cora, and Nando De Freitas. A Tutorial on Bayesian Optimization of Expensive Cost Functions , with Application to Active User Modeling and Hierarchical Reinforcement Learning. *Arxiv preprint arXiv:1012.2599*, 2010.

[73] M. Ciocarlie, C. Goldfeder, and P. Allen. Dimensionality reduction for hand-independent dexterous robotic grasping. In *Proc. of IROS 2007*, 2007.

[74] Rosen Diankov. Openrave: A planning architecture for autonomous robotics. *Robotics Institute, Pittsburgh, PA, Tech. Rep.*, (July), 2008.

[75] Daniel E Finkel. DIRECT Optimization Algorithm User Guide. 2003.

[76] Marcus Frean and Phillip Boyle. Using Gaussian processes to optimize expensive functions. *AI 2008: Advances in Artificial Intelligence*, 2008.

[77] DR Jones, Matthias Schonlau, and WJ Welch. Efficient global optimization of expensive black-box functions. *Journal of Global optimization*, pages 455–492, 1998.

[78] O. B. Kroemer, R. Detry, J. Piater, and J. Peters. Combining active learning and reactive control for robot grasping. *Robot. Auton. Syst.*, 58(9):1105–1116, 2010.

[79] H. Liu, X. Song, J. Bimbo, K. Althoefer, and L. Seneviratne. Intelligent Fingertip Sensing for Contact Information Identification. In *ASME/IFToMM Int. Conf. Reconfigurable Mechanisms and Robots*, 2012.

[80] D. Lizotte. Practical bayesian optimization. Technical Report PhD Thesis, university of Alberta, Edmonton, Alberta, Canada, November 2008.

[81] Daniel Lizotte, Tao Wang, Michael Bowling, and Dale Schuurmans. Automatic gait optimization with gaussian process regression. *Proc. of IJCAI*, 2007.

[82] Ruben Martinez-cantin, De Freitas Nando, Eric Brochu, José Castellanos, and Arnaud Doucet. A Bayesian exploration-exploitation approach for optimal online sensing and planning with a visually guided mobile robot. *Autonomous . . .* , (2005), 2009.

[83] A. Miller, S. Knoop, H. Christensen, and P. Allen. Automatic grasp planning using shape primitives. In *Proc. of ICRA 2003*, 2003.

[84] A. Morales, T. Asfour, P. Azad, S. Knoop, and R. Dillmann. Integrated grasp planning and visual object localization for a humanoid robot with five-fingered hands. In *Proc. of IROS 2006*, 2006.

[85] Z. Li R. Murray and S. Sastry. *A Mathematical Introduction to Robotic Manipulation*. CRC Press, Boca Raton, FL, 1994.

[86] C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning*. The MIT Press, 2006.

[87] Máximo a. Roa and Raúl Suárez. Finding locally optimum force-closure grasps. *Robotics and Computer-Integrated Manufacturing*, 25(3):536–544, June 2009.

[88] N. Vahrenkamp, T. Asfour, and R. Dillmann. Simultaneous grasp and motion planning: humanoid robot ARMAR-III. *IEEE Robotics and Automation Magazine*, 19(2):43–57, 2012.

[89] Paul Baxter and Will Browne. Memory-based cognitive framework: A low-level association approach to cognitive architectures. In George Kampis, István Karsai, and Eörs Szathmáry, editors, *Advances in Artificial Life. Darwin Meets von Neumann*, volume 5777 of *Lecture Notes in Computer Science*, pages 402–409. Springer Berlin / Heidelberg, 2011.

[90] Erik Billing and Thomas Hellström. A formalism for learning from demonstration. *Paladyn. Journal of Behavioral Robotics*, 1:1–13, 2010.

[91] Erik Billing, Thomas Hellström, and Lars-Erik Janlert. Behavior recognition for learning from demonstration. In *IEEE International Conference on Robotics and Automation*, pages 866–872, 2010.

[92] S. Bodiroza, G. Schillaci, and V.V. Hafner. Robot ego-sphere: An approach for saliency detection and attention manipulation in humanoid robots for intuitive interaction. In *IEEE-RAS International Conference on Humanoid Robots*, pages 689–694, 2011.

[93] D. Caligiore, T. Ferrauto, D. Parisi, N. Accornero, M. Capozza, and G Baldassarre. Using motor babbling and hebb rules for modeling the development of reaching with obstacles and grasping. In *International Conference on Cognitive Systems*, 2008.

[94] Gang Cheng, Norman Hendrich, and Jianwei Zhang. In-hand manipulation action gist extraction from a data-glove. In *IEEE International Conference on Cognitive Systems and Information Processing*, 2012.

[95] A.L. Ciancio, L. Zollo, E. Guglielmelli, D. Caligiore, and G. Baldassarre. Hierarchical reinforcement learning and central pattern generators for modeling the development of rhythmic manipulation skills. In *IEEE International Conference on Development and Learning*, pages 1–8, 2011.

[96] Pinar Civicioglu and Erkan Besdok. A conceptual comparison of the cuckoo-search, particle swarm optimization, differential evolution and artificial bee colony algorithms. *Artificial Intelligence Review*, pages 1–32, 2011.

[97] Craig Corcoran and Robert Jr Platt. A measurement model for tracking hand-object state during dexterous manipulation. In *IEEE International Conference on Robotics and Automation*, pages 4302–4308, 2010.

[98] Marc Peter Deisenroth, Carl Edward Rasmussen, and Dieter Fox. Learning to control a low-cost manipulator using data-efficient reinforcement learning. In *Robotics: Science and Systems*, 2011.

[99] Yiannis Demiris and Anthony Dearden. From motor babbling to hierarchical learning by imitation: A robot developmental pathway. In *International Workshop on Epigenetic Robots*, pages 31–37, 2005.

[100] Yiannis Demiris and Andrew Meltzoff. The robot in the crib: A developmental analysis of imitation skills in infants and robots. *Infant Child Development*, 17(1):43–53, 2008.

[101] Guanglong Du, Ping Zhang, Jianhua Mai, and Zeling Li. Markerless kinect-based hand tracking for robot teleoperation. *International Journal of Advanced Robotic Systems*, 9(36):1–10, 2012.

[102] Diego R Faria, Ricardo Martins, Jorge Lobo, and Jorge Dias. Manipulative tasks identification by learning and generalizing hand motions. In *International Federation for Information Processing*, 2011.

[103] Michele Folgheraiter, Ilario Baragiola, and Giuseppina Gini. *Teaching grasping to a humanoid hand as a generalization of human grasping data*, chapter Knowledge Exploration in Life Science Informatics, pages 139–150. 2004.

[104] X H Gao, M H Jin, , L Jiang, Z W Xie, P He, L Yang, Y W Liu, R Wei, H G Cai, H Liu, J Butterfass, M Grebenstein, N Seitz, and G Hirzinger. The hit/dlr dexterous hand: Work in progress. In *IEEE International Conference on Robotics and Automation*, pages 3164–3168, 2003.

[105] Abhinav Gupta, Aniruddha Kembhavi, and Larry Davis. Observing human-object interactions: Using spatial and funcitional compatibility for recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(10):1775–1789, 2009.

[106] Donald Olding Hebb. *The organization of behavior: a neuropsychological theory*. New York: Wiley & Sons, 1949.

[107] Tatsuya Ishihara, Akio Namiki, Masatoshi Ishikawa, and Makoto Shimojo. Dynamic pen spinning using a high-speed multifingered hand with high-speed tactile sensor. In *IEEE-RAS International Conference on Humanoid Robots*, pages 258–263, 2006.

[108] J Kennedy and R Eberhart. Particle swarm optimization. In *IEEE International Conference on Neural Networks*, pages 1942–1948, 1995.

[109] S Mohammad Khansari-Zadeh and Aude Billard. Learning stable nonlinear dynamical systems with gaussian mixture models. *IEEE Transactions on Robotics*, 27(5):943–957, 2011.

[110] Hedvig Kjellstroem, Javier Romero, David Martinez, and Danica Kragic. Simultaneous visual recognition of manipulation actions and manipulated objects. In *European Conference on Computer Vision*, 2008.

[111] Y. Kobayashi and S. Hosoe. Planning-space shift learning: Variable-space motion planning toward flexible extension of body schema. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3107–3114, 2009.

[112] Yuichi Kobayashi and Shigeyuki Hosoe. Planning-space shift motion generation: Variable-space motion planning toward flexible extension of body schema. *Journal of Intelligent and Robotic Systems*, 62:467–500, 2011.

[113] Manuel Lopes, Francisco Melo, Luis Montesano, and José Santos-Victor. Abstraction levels for robotic imitation: Overview and computational approaches. In Olivier Sigaud and Jan Peters, editors, *From Motor Learning to Interaction Learning in Robots*, volume 264 of *Studies in Computational Intelligence*, pages 313–355. Springer Berlin / Heidelberg, 2010.

[114] C S Lovchik and M A Diftler. The robonaut hand: A dexterous robot hand for space. In *IEEE International Conference on Robotics and Automation*, pages 907–912, 1999.

[115] Jeremy Maitin-Shepard, Marco Cusumano-Towner, Jinna Lei, and Pieter Abbeel. Cloth grasp point detection based on multiple-view geometric cues with application to robotic towel folding. In *IEEE International Conference on Robotics and Automation*, 2010.

[116] Andrew N. Meltzoff and Keith M. Moore. Explaining facial imitation: a theoretical model. *Early Development and Parenting*, 6(3-4):179–192, 1997.

[117] Giorgio Metta and Paul Fitzpatrick. Better vision through manipulation. *Adaptive Behavior*, 11(2):109–128, 2003.

[118] Satoru Mizusawa, Akio Namiki, and Masatoshi Ishikawa. Tweezers type tool manipulation by a multifingered hand using a high-speed visual servoing. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2709–2714, 2008.

[119] L. Montesano, M. Lopes, A. Bernardino, and J. Santos-Victor. Affordances, development and imitation. In *IEEE International Conference on Development and Learning*, pages 270–275, 2007.

[120] Lael U Odhner and Aaron M Dollar. Dexterous manipulation with underactuated elastic hands. In *IEEE International Conference on Robotics and Automation*, 2011.

[121] Dimitri Ognibene, Angelo Rega, and Gianluca Baldassarre. A model of reaching that integrates reinforcement learning and population encoding of postures. In *International Conference on From Animals to Animats: Simulation of Adaptive Behavior*, pages 381–393, 2006.

[122] E. Oztop and M.A. Arbib. A biologically inspired learning to grasp system. In *International Conference of the IEEE Engineering in Medicine and Biology Society*, pages 857–860, 2001.

[123] Nancy S Pollard and Jessica K Hodgins. Generalizing demonstrated manipulation tasks. In *Workshop on the Algorithmic Foundations of Robotics*, 2002.

[124] M. Rolf, J.J. Steil, and M. Gienger. Goal babbling permits direct learning of inverse kinematics. *IEEE Transactions on Autonomous Mental Development*, 2(3):216–229, 2010.

[125] Ryo Saegusa, Giorgio Metta, Giulio Sandini, and Sophie Sakka. Active motor babbling for sensorimotor learning. In *IEEE International Conference on Robotics and Biomimetics*, pages 794–799, 2008.

[126] Anis Sahbani, Jean-Philippe Saut, and Veronique Perdereau. An efficient algorithm for dexterous manipulation planning. In *IEEE International Multi-Conference on Systems, Signals and Devices*, 2007.

[127] Y. Shi and R.C. Eberhart. A modified particle swarm optimizer. In *IEEE International Conference on Evolutionary Computation*, pages 69–73, 1998.

[128] Aaron P. Shon, Storz Joshua J., Andrew N. Meltzoff, and Rajesh P. N. Rao. A cognitive model of imitative development in humans and machines. *International Journal of Humanoid Robotics*, 4(2):387–406, 2007.

[129] Peter Stone, Patrick Beeson, Tekin Mericli, and Ryan Madigan. Austin robot technology. Technical report, University of Texas at Austin, 2007.

[130] Yan Wu and Yiannis Demiris. Towards one shot learning by imitation for humanoid robots. In *IEEE International Conference on Robotics and Automation*, 2010.

[131] Yuji Yamakawa, Akio Namiki, and Masatoshi Ishikawa. Card manipulation using a high-speed robot system with high-speed visual feedback. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4762–4767, 2012.

[132] Hanna Yousef, Mehdi Boukallel, and Kaspar Althoefer. Tactile sensing for dexterous in-hand manipulation in robotics — a review. *Sensors and Actuators A: Physical*, 167(2):171–187, 2011.

[133] HANDLE project, deliverable D3, *Instrumented sensing objects*, www.handleproject.eu, 2009

[134] HANDLE project, deliverable D4, *Protocol for the corpus of sensed grasp and handling data*, www.handleproject.eu, 2009

[135] HANDLE project, deliverable D5, *Sensor system specification and evaluation of different methods for object recognition*, *Protocol for the corpus of sensed grasp and handling data*, www.handleproject.eu, 2009

[136] HANDLE project, deliverable D6, *Tactile sensing for data-gloves*, www.handleproject.eu, 2009

[137] HANDLE project, deliverable D7, *Algorithms for real-time collision avoidance accompanied by a report on existing planning methods*, www.handleproject.eu, 2009

[138] HANDLE project, deliverable D9, *Second robot hardware platform*, www.handleproject.eu, 2010

[139] HANDLE project, deliverable D10, *Annotated cataloge of grasp and force/-motion signatures*, www.handleproject.eu, 2010

[140] HANDLE project, deliverable D12, *Hand-state models of human grasping and manipulation skills*, www.handleproject.eu, 2010

[141] HANDLE project, deliverable D13, *Algorithms for planning the grasping of objects for manipulation and for planning the in-hand manipulation*, www.handleproject.eu, 2011

[142] HANDLE project, deliverable D14, *Improving known actions from motor babbling*, www.handleproject.eu, 2011

[143] HANDLE project, deliverable D15, *Reports on organised workshops with generated documentation*, www.handleproject.eu, 2011

[144] HANDLE project, deliverable D16, *Hand design report*, www.handleproject.eu, 2011

[145] HANDLE project, deliverable D17, *Automatic dataset reduction system for grasp motion data*, www.handleproject.eu, 2011

[146] HANDLE project, deliverable D18, *Visual and tactile perception algorithms for grasping*, www.handleproject.eu, 2011

[147] HANDLE project, deliverable D19, *Embedded electronic design report*, www.handleproject.eu, 2011

[148] HANDLE project, deliverable D20, *Skin design report*, www.handleproject.eu, 2011

[149] HANDLE project, deliverable D21, *Motion primitives for human-like grasping and tool use with a robotic hand*, www.handleproject.eu, 2012

[150] HANDLE project, deliverable D22, *Low-level controllers desgin including hybrid force/position control and visual servoing*, www.handleproject.eu, 2012

[151] HANDLE project, deliverable D23, *Visual and tactile perception system evaluation report*, www.handleproject.eu, 2012

[152] HANDLE project, deliverable D24, *Parameterizing and creating new actions*, www.handleproject.eu, 2012

[153] HANDLE project, deliverable D25, *Complete anthropomorphic hand*, www.handleproject.eu, 2012

[154] HANDLE project, deliverable D30, *Discovering new affordances from behavioral babbling*, www.handleproject.eu, 2013

[155] HANDLE project, deliverable D31, *Developmental methods in exploration learning*, www.handleproject.eu, 2013

[156] HANDLE project, extra deliverable, *Overview of the ROS software*, www.handleproject.eu, 2013.