Project Acronym:	GRASP
Project Type:	IP
Project Title:	Emergence of Cognitive Grasping through Introspection, Emulation and Surprise
Contract Number:	215821
Starting Date:	01-03-2008
Ending Date:	28-02-2012

Deliverable Number:	D19
Deliverable Title :	Vocabulary of human and robot actions
Type (Internal, Restricted, Public):	PU
Authors	D. Kragic, D. Song, J. Bohg, T. Asfour, R. Dillmann, M. Do
Contributing Partners	KTH, UniKarl

Contractual Date of Delivery to the EC:	28-02-2011
Actual Date of Delivery to the EC:	28-02-2011

Contents

Chapter 1

Executive summary

Deliverable D19: *Vocabulary of human and robot actions* presents the third year developments within WP2 - "Representations and Ontology for learning and Abstraction of Grasping". According to the Technical Annex, deliverable D19 presents the activities in the context of Tasks 2.1-2.3:

- [**Task 2.1**] - Definition of the ontology: definition of sensory-motor control for action and object-action learning
- [**Task 2.2**] - Vocabulary of human and robot actions/interactions
- [**Task 2.3**] - Evaluation of representation: Evolving ontology through modeling of the perception-action cycle

The work in this deliverable relates to the following third year Milestones:

- [**Milestone 7**] - Observing consequences of grasping; vocabulary of robot action/interactions and definition of a hierarchical structure of features.
- [**Milestone 9**] - Integrating contextual representation in the ontology and development of the attention system with view planning; implementation on active head.

The progress in WP2 is presented in the below summarized scientific publications, attached to this deliverable.

- In Attachment A we present work on pre-grasp planning. In manipulation tasks that require object acquisition, pre-grasp interaction such as sliding adjusts the object in the environment before grasping. This change in object placement can improve grasping success by making desired grasps reachable. However, the additional sliding action prior to grasping introduces more complexity to the motion planning process, since the hand pose relative to the object does not need to remain fixed during the pre-grasp interaction. Furthermore, anthropomorphic hands in humanoid robots have several degrees of freedom that could be utilized to improve the object interaction beyond a fixed grasp shape. We present a framework for synthesizing pre-grasp interactions for high-dimensional anthropomorphic manipulators. The planning is tractable because information from pre-grasp manipulation examples narrows the search to promising hand poses and shapes. In particular, we show the value of organizing the example data according to object category templates. The template information further focuses the search based on the object features, which increases the success of adapting a template pose and decreases the planning time.
- In Attachment B, we present a grasp representation in task space exploiting position information of the fingertips. We propose a new way for grasp representation in the task space, which provides a suitable basis for grasp imitation learning. Inspired by neuroscientific findings, finger movement synergies in the task space together with fingertip positions are used to derive a parametric low-dimensional grasp representation. Taking into account correlating finger movements, we describe grasps using a system of virtual springs to connect the fingers, where different grasp types are defined by parameterizing the spring constants. Based on such continuous parameterization, all instantiation of grasp types and all hand preshapes during a grasping action (reach, preshape, enclose, open) can be represented. We present experimental results, in which the spring constants are merely estimated from fingertip motion tracking using a stereo camera setup of a humanoid robot. The results show that the generated grasps based on the proposed representation are similar to the observed grasps.

- In Attachment C, we study visual servoing in a framework of detection and grasping of unknown objects. Classically, visual servoing has been used for applications where the object to be servoed on is known to the robot prior to the task execution. In addition, most of the methods concentrate on aligning the robot hand with the object without grasping it. In our work, visual servoing techniques are used as building blocks in a system capable of detecting and grasping unknown objects in natural scenes. We show how different visual servoing techniques facilitate a complete grasping cycle.

- Attachment D, present the work related to Task 2.2 and Task 2.3. In year two, Attachment D in deliverable D12, our work was limited in an expert-fixed network structure for BNs, and the training data was in the original space of the modeled variables. These constrains BNs from modeling large range of sensory streams, and scaling to a large number of objects and manipulation tasks. These problems motivated the work for the third year. Here, we adopted a Gaussian process based latent variable model (GPLVM) to specifically explore the possible latent structure of the task-related sensory input. The aim is to select a reduced set of variables that most efficiently disambiguates the tasks, and at the same time allow accurate reconstruction of individual features. Such a reduced set of variables can then be used to efficiently construct and train the BN-based task constraint models.

- Attachment E, presents the work related to Task 2.2 and Task 2.3. The work introduces a novel multivariate discretization approach based on sparse, non-parametric, probabilistic dimensionality reduction. The new approach opens up the possibility of using Bayesian networks (BN) for scenarios where the complexity of the data have previously made BN inapplicable. In robot grasping, a major challenge in modeling with BNs is learning the structure from both discrete and multivariate continuous data. A common approach in such situations is to discretize continuous data before structure learning. However efficient methods to discretize high-dimensional variables are largely lacking. This paper presents a novel method specifically aiming at discretization of high-dimensional, high-correlated data. The model is fully probabilistic and capable to facilitate structure learning from discretized data, while at the same time retain the continuous representation. Compared with traditional discretization schemes, our model excels both in task classification and prediction of hand grasp configurations.

Appendix A

Attached Papers

[A] **Templates for pre-grasp interaction**; L. Chang, D. Kappler, N.S. Pollard, T. Asfour, R. Dillmann; Robotics and Autonomous Systems, submitted

[B] **Towards a Unifying Grasp Representation for Imitation Learning on Humanoid Robots**; M. Do, T. Asfour, R. Dillmann; IEEE International Conference on Robotics and Automation, IEEE ICRA; Shanghai, 2011.

[C] **Visual Servoing on Unknown Objects**; X. Gratal, J. Romero, J. Bohg and Danica Kragic; Journal of Mechatronics, (submitted)

[D] **Task modeling in imitation learning using latent variable models**, C.-H. Ek, D. Song, K. Huebner and D. Kragic, 10th IEEE-RAS International Conference on Humanoid Robots, Nashville, TN, US, December 2010.

[E] **Multivariate Discretization for Bayesian Network Structure Learning in Robot Grasping**; C.-H. Ek, D. Song, K. Huebner and D. Kragic; IEEE International Conference on Robotics and Automation, IEEE ICRA; Shanghai, 2011.

Elsevier Editorial System(tm) for Robotics and Autonomous Systems
Manuscript Draft

Manuscript Number:

Title: Templates for pre-grasp interaction

Article Type: SI: IROS' 2010

Keywords: pre-grasp interaction; object manipulation; humanoid robotics; pushing; sliding

Corresponding Author: Ms. Lillian Chang,

Corresponding Author's Institution: Intel Labs Seattle

First Author: Lillian Chang

Order of Authors: Lillian Chang; Daniel Kappler; Nancy S Pollard, Ph.D.; Tamim Asfour, Dr.-Ing.; Rudiger Dillmann, Prof. Dr.-Ing.

Abstract: In manipulation tasks that require object acquisition, pre-grasp interaction such as sliding adjusts the object in the environment before grasping. This change in object placement can improve grasping success by making desired grasps reachable. However, the additional sliding action prior to grasping introduces more complexity to the motion planning process, since the hand pose relative to the object does not need to remain fixed during the pre-grasp interaction. Furthermore, anthropomorphic hands in humanoid robots have several degrees of freedom that could be utilized to improve the object interaction beyond a fixed grasp shape.
We present a framework for synthesizing pre-grasp interactions for high-dimensional anthropomorphic manipulators. The planning is tractable because information from pre-grasp manipulation examples narrows the search to promising hand poses and shapes. In particular, we show the value of organizing the example data according to object category templates. The template information further focuses the search based on the object features, which increases the success of adapting a template pose and decreases the planning time.

***Research Highlights**

Templates for pre-grasp interaction
submission to RAS Special Issue on Autonomous Grasping

Research Highlights
==========

We present a framework and a system implementation for robot manipulation of objects on a flat surface. Our method incorporates the advantages of pre-grasp interaction with direct grasping. The primary parts of our framework are the organization of examples in object-based template categories and the representation of the hand poses by components which can be modified to adapt to new object scaling and rotation.

We show that our method makes pre-grasp interaction planning tractable by quickly narrowing the set of examples to promising templates for adaptation. Our data is based on examples observed in a human motion capture study. Our validation includes both simulation experiments with our proposed framework as well as robustness testing for variations without template organization or template mis-classification. In addition, we show that the generated plans are physically plausible with physical demonstration on multiple humanoid robots. We also introduce some evidence that the pre-grasp strategy is more natural looking with a user perception evaluation.

Templates for pre-grasp sliding interactions

Daniel Kappler[c,2,**], Lillian Y. Chang[a,b,1,*], Nancy S. Pollard[b,1,3], Tamim Asfour[c,4]

[a]*Intel Labs Seattle, Seattle, Washington, United States*
[b]*Robotics Institute, School of Computer Science, Carnegie Mellon University, Pittsburgh, Pennsylvania, United States*
[c]*Institute for Anthropomatics, Karlsruhe Institute of Technology, Karlsruhe, Germany*

Abstract

In manipulation tasks that require object acquisition, pre-grasp interaction such as sliding adjusts the object in the environment before grasping. This change in object placement can improve grasping success by making desired grasps reachable. However, the additional sliding action prior to grasping introduces more complexity to the motion planning process, since the hand pose relative to the object does not need to remain fixed during the pre-grasp interaction. Furthermore, anthropomorphic hands in humanoid robots have several degrees of freedom that could be utilized to improve the object interaction beyond a fixed grasp shape. We present a framework for synthesizing pre-grasp interactions for high-dimensional anthropomorphic manipulators. The planning is tractable because information from pre-grasp manipulation examples narrows the search to promising hand poses and shapes. In particular, we show the value of organizing the example data according to object category templates. The template information further focuses the search based on the object features, which increases the success of adapting a template pose and decreases the planning time.

Keywords:
pre-grasp interaction, object manipulation, humanoid, pushing, sliding

1. Introduction

In service tasks involving object fetching or transport, an autonomous manipulator must acquire the object before delivery. When the desired contact surfaces for grasping are within reach, a direct reach-to-grasp motion is sufficient to achieve object acquisition. However, in unstructured environments such as the home or office, object placement may change day-to-day and not always convenient for reaching the desired grasp. In these scenarios, *pre-grasp interaction* with a movable object can adjust the object placement to improve the reachability of good grasps. Human examples of pre-grasp interaction include sliding flat objects such as a credit card to a table edge to grasp it, pushing a heavy box near the body mass center for easier lifting, or rotating a handled object such as a water pitcher.

In this paper, we present a method for synthesizing pre-grasp sliding interactions for an anthropomorphic manipulator. Many service robots have a humanoid form designed to perform manipulation tasks with human-like motions [1–3]. The multi-fingered hands of these robots have several degrees of freedom for achieving a wide range of hand shapes to accommodate different object geometries. However, the manipulator's kinematic freedoms as well as additional object motion introduce more complexity to the planning process for pre-grasp interactions.

To make the planning tractable, our framework makes use of human pre-grasp manipulation examples to narrow the search to promising hand poses for the pushing interaction. We have observed that in human pre-

*Corresponding author
**Principal corresponding author
Email addresses: daniel.kappler@kit.de (Daniel Kappler), lillian.chang@intel.com (Lillian Y. Chang), nsp@cs.cmu.edu (Nancy S. Pollard), asfour@kit.edu (Tamim Asfour)

[1]L. Y. Chang is an NSF/CRA/CCC Computing Innovation Fellow. L. Y. Chang also received support from a NASA Harriet G. Jenkins Pre-Doctoral Fellowship while at Carnegie Mellon University.
[2]D. Kappler received support from the InterACT Exchange project at Carnegie Mellon University.
[3]This work was also supported by the National Science Foundation (CCF-0702443).
[4]The work described in this paper was partially conducted within the EU Cognitive Systems project GRASP (IST-FP7-IP-215821) funded by the European Commission and the German Humanoid Research project SFB588 funded by the German Research Foundation (DFG: Deutsche Forschungsgemeinschaft).

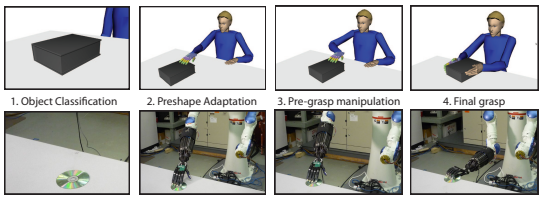


Figure 1: Overview of the pre-grasp strategy, where an object is slid on the table before the final grasp.

grasp interaction of objects on a tabletop, the manipulation tended to include pre-grasp sliding toward the body if the object is out of reach or inconvenient to grasp. These sliding manipulations also tended to exhibit similar hand shapes and poses relative to the object according to object categories based on object shape and weight. These patterns form the basis of our framework, where pre-grasp interaction examples are organized into templates based on object categories.

In our framework, example hand preshapes for pre-grasp sliding are stored in the example database according to object category to retain the context information of the template. The information stored for one object template includes the hand preshape—composed of both the starting hand pose relative to the object and the finger joint configurations, the associated preshapes suitable for the final grasp to achieve object acquisition, and the constraints on the related pre-grasp interaction strategies. Constraints on the pre-grasp interaction strategies include context information such as final goal regions for the object relocation, as well as constraints on the object motion or robot joint motion.

These preshape templates enable efficient automatic generation of hand poses for pre-grasp object interaction, which ultimately improve the success of grasping for object acquisition. In particular, we show the value of organizing the example data according to object category templates. The template information further focuses the search based on the object features, which increases the success of adapting a template pose and decreases the planning time.

In the following, we first review related work on grasping templates and manipulation planning. We then describe the collection of human pre-grasp interaction examples. The remainder of the paper presents our template-based framework for planning pre-grasp sliding motions, our experimental results and validation tests, and a discussion of future steps to generalize this framework for an even broader set of manipulation actions.

2. Related Work

Our framework for planning pre-grasp interactions is composed of four main phases (Fig. 1): (1) object classification to determine the template category, (2) adaptation and evaluation of candidate hand preshapes from the initial template, (3) simulation of the pre-grasp interaction that adjusts object placement, and (4) planning the final grasp which acquires the object. We describe here the previous work related to the different components of our framework.

The first phase of object classification determines a category based on object features such as shape and weight. This category narrows the set of candidate hand preshapes to those associated with similar objects. Previous investigation of human grasps has explored how a set or taxonomy of prototypical hand shapes can describe the space of hand configurations for grasping tasks [4–6]. In imitation learning, several researchers have presented methods of learning the classification of a human demonstration of a grasp, which can then be mapped to a robot hand configuration [7–11]. In our work, the classification is on object features rather than the hand shape itself. In this regard, it is most similar to the learning methods in [12–14] where features of an object's component sub-shapes are used to learn which sub-shape is a handle or suitable handshapes for grasping the sub-shape. Our method does not use object decomposition for determining grasp contacts. The contact surfaces on the object are determined in later phases of our framework based on the preshape examples associated with each object category.

Both the second phase of preshape adaptation and the fourth phase of final grasp planning involve modification of template hand shapes to fit a new object geometry. A method to refine a prototype hand shape has been developed [15] to fit a static hand preshape to the surface of a new object. Our method for adapting hand configurations for pre-grasp interactions is similar in the adjustment of the finger joints to achieve more contact with the object. In our framework, though, the preshape adaptation is evaluated for non-grasping (non-prehensile) hand poses by how well the contact forces contribute to the desired object adjustment in pre-grasp interaction. Other methods of grasp template refinement to new object shapes have also been developed for synthesizing new contact points on the object surface [16, 17], from image analysis instead of 3-D geometry [18], and grasp synergy subspaces [19].

Grasp synthesis without template context has also been investigated from several aspects. Berenson et al. [20] proposed a method to precompute a set of possible

grasps offline for later online evaluation against environmental constraints. Przybylski et al. suggested in [21] to use the medial axis to reduce the number of possible grasp candidates. Finding model-based analytical grasps is discussed by Bicchi and Kumar in [22]. For evaluating the stability of a possible lifting grasp, Ferrari and Canny [23] proposed the force closure metric, which measures the force exerted on the object at the contact points. More recent research about grasp quality is presented by Miller and Allen [24].

The third phase of planning the object interaction motion is related to the field of manipulation planning. There are several strategies of pre-grasp interaction manipulation for adjust the object on the support surface. Our framework is intentionally designed to accommodate multiple modes of interaction, and this paper primarily discusses pre-grasp interaction by planar pushing as a widely applicable action. The simplest form of pushing is single-freedom reorientation in the plane, which has been used in pre-grasp rotation to grasp hard-to-reach object handles [25–27]. Planning techniques for more general pushing and sliding actions using non-prehensile manipulation are discussed in [28–30]. Recent work [31] uses a short push or push-grasp as a type of pre-grasp action primitive for bringing objects into the hand after a reaching motion. Toppling as well as tumbling presented by Lynch et al. [32] are also possible pre-grasp manipulations. For humanoid robots, multi-modal interaction combining locomotion and object pushing has been been developed by [33]. A whole-body manipulation strategy for pivoting large, heavy objects has been presented by [34] as the primary manipulation task that avoids completely grasping or lifting the object.

Our presented framework also builds upon multiple concepts of motion planning for synthesizing arm actions. Our templates for pre-grasp interaction hand shapes store context information about the hand shape and the hand pose relative to the object. The examples do not include arm posture configurations, which would be specific to a particular base placement of the robot relative to the desired hand pose. Thus, it is necessary to not only synthesize hand shapes but also collision-free arm postures to reach the desired hand pose. Many humanoid robots have at least 7-DoF arms which leads to redundant inverse kinematics problems to achieve hand poses. This is an extensively studied topic in the computer graphics and robotics community [35–37]. These approaches enable computation of manipulator configurations for grasping, sliding and rotating manipulations. For a complete motion, a path between the initial pose and the calculated one has to be planned. Additional constraints for such a path are obstacle avoidance and joint limitations. Latombe [38] discusses traditional approaches to this problem and Rapidly-exploring Random Tree (RRT) methods by Lavalle [39].

Current methods [13][20][21] for planning object fetching search direct grasp solutions based on a known object pose. Our pre-grasp strategy augments such methods by a preliminary step which reconfigures the object pose through a suitable pre-grasp manipulation actions. This strategy not only results in higher success rates for finding stable grasp solutions, but it also increases the final grasp stability for grasping. Also, even when a direct grasp solution is available, our approach enables faster online planning based on the focused search from context knowledge.

3. Human examples of pre-grasp interaction

In many scientific fields, successful techniques have been inspired by studying solutions provided by nature. Pre-grasp interaction is a huma-inspired manipulation strategy that has been observed in many typical household activities [40]. Previously, detailed study [41] of a specific type of interaction, pre-grasp rotation, has led to the development of similar techniques for object orientation for robot manipulators [27, 42].

In this work, we use examples from human actions for more general interactions during the grasping process, not just pre-grasp rotation. Our initial survey provided insight into possible underlying patterns for pre-grasp interaction with objects on a cluttered surface. Our follow-up observations instrumented the objects and people to capture specific examples of their hand motions relative to the object. These examples form the basis of our database for preshapes templates for the pre-grasp interaction.

Both studies of human pre-grasp interaction were conducted with participants who provided their voluntary consent, in accordance with the Carnegie Mellon Institutional Review Board policies.

3.1. Video survey

We first conducted an informal video survey to investigate different human strategies and possible object-based patterns. For each participant various objects such as a stapler, books, and CDs were randomly placed on a table. The participants were instructed to remove the objects from the table and place them on a nearby chair. They sat at the table while performing the task and were permitted to use only one hand. The major strategy we observed was that people tend to perform pre-grasp manipulation, especially sliding the object on the table, to

pick up the object. Furthermore, they used similar hand poses for pre-grasp manipulation when grasping objects of similar shapes.

3.2. Motion capture

We then ran a motion capture experiment to record two sets of data from human examples of pre-grasp manipulation:

1. the hand positions relative to the object and
2. the hand preshape configuration,

both just before contact with the object. We expected that both the position and preshape will change for different object categories and different environmental conditions. By environmental conditions, we mean the object pose relative to the robot pose.

The setup consisted of a table with randomly-placed objects, and the participants were instructed to move all objects to another table. The setup required participants to walk few meters between the tables, which was intended to implicitly force the use of stable object grasps for acquisition. Unlike the video survey, there was no constraint for one-handed grasps in this experiment, because we preferred to capture the natural grasping behavior as much as possible.

We captured examples of hand positions and configurations from 4 adult participants. In some cases, the participants held multiple objects in one hand while acquiring the next object with the other hand. We hypothesize this may be an effort to reduce the number of walks to the second table. All participants used pre-grasp manipulation to slide or reorient a few objects, and three used it for most objects. The grasp shapes and positioning on the object were similar between different participants.

4. Pre-grasp strategy

We present a data-driven strategy which is designed to automatically perform pre-grasp manipulation actions to fetch an object from a surface. That is, given an object in an initial pose on a surface, our method plans a solution for a robot to adjust the object to a final pose, within a distinct final region, using a suitable pre-grasp manipulation strategy. This data-driven approach is based on context knowledge consisting of look-up structures for object categories, hand configurations and poses, discrete actions, constraints, goal regions, and grasp-types which are described in the following sections.

4.1. Representations

At the high level, the context knowledge is organized by *object categories* $\mathbb{O}$. For each object category, there are multiple entries for different pre-grasp manipulation contexts which allow successful pre-grasp manipulation for a given object within this category with a high probability. The context per object category consists of two parts, a set of *preshapes* $\mathbb{S}$ and a set of *pre-grasp manipulation* data structures $\mathbb{M}$.

$$\mathbb{O} = (\mathbb{S}, \mathbb{M}) \tag{1}$$

The next sections describe the *preshape* $\mathbb{S}$ and *pre-grasp manipulation* $\mathbb{M}$ data structure.

4.1.1. Preshape

The idea behind the *preshape* data structure is, that a hand configuration in a distinct pose relative to an object, used for pre-grasp manipulation actions, can be efficiently adapted to other objects within the same category to perform the similar actions. Hence, only a small set of example preshapes is needed to be able to find suitable hand configuration and poses to perform pre-grasp manipulation actions for an object. In that context, we introduce the *preshape* data structure $\mathbb{S}$ as:

$$\mathbb{S} = (\mathbf{c}, P, C, \mathbf{G}, \mathbb{M}) \tag{2}$$

Every preshape provides a hand configuration $\mathbf{c}$, which is defined by the joint values of the given robot hand. A preshape also supplies a set of starting poses P, which describe the hand position and orientation. This set P should be invariant in terms of rotation and scaling for objects within the same category. Therefore, we propose to compute P at runtime to provide appropriate starting poses $\mathbf{p}$ with respect to a given object. Hence, different preshape definitions are likely for different or even the same pre-grasp manipulation actions. To be able to perform the pre-grasp manipulations $\mathbb{M}$ possible for a preshape, the hand configuration $\mathbf{c}_a$ and poses P_a adapted to the object have to satisfy constraints C, for example finger contact with the object or mechanical joint limits. Additionally, the selection of a distinct preshape permits the selection of subset of grasp-types $\mathbf{G}$ available by the robot platform which are reasonable for final grasping.

4.1.2. Pre-grasp manipulation

The goal for pre-grasp manipulation strategy is to relocate an object into a final region where the object is

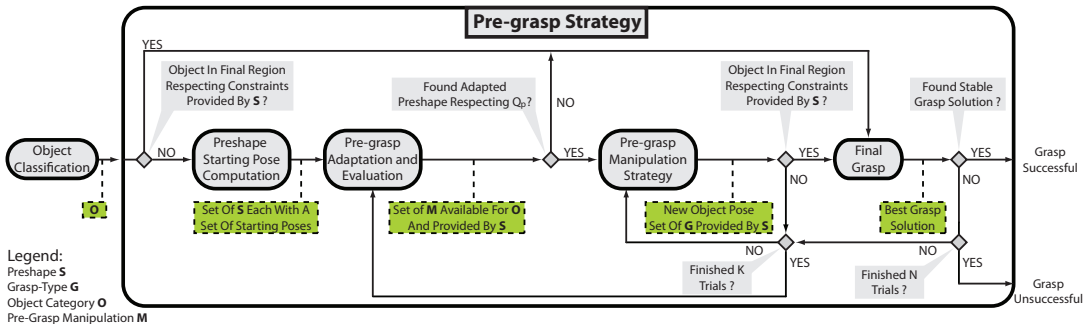


Figure 2: The proposed pre-grasp strategy architecture with the four phases: "Preshape Starting Pose Computation, Pre-grasp Adaptation and Evaluation, Pre-grasp Manipulation, Final Grasp" embedded in the grasp planning context.

more likely to be successfully grasped. For that purpose, we propose the pre-grasp manipulation $\mathbb{M}$ data structure:

$$\mathbb{M} = (a, C, F) \tag{3}$$

A pre-grasp manipulation is defined by an action a, which in this context may be for example toppling, tumbling, rotating, pushing, and sliding. In general an action a has to satisfy constraints C during manipulation, for example constant object contact, force limitations to the robot joints as well as to the object surface, or object orientations like ensuring that a cup is not spilling. Additionally for particular grasp-types $\mathbf{G}$ and object categories $\mathbb{O}$ there are constraints C for the object within the final region F such as whether the handle of a pan is reachable. The final region F is defined by the intersection of the region in which grasp-types G, provided by the selected preshape $\mathbb{S}$, are feasible in the current robot workspace, the surface region, and the region an action is likely to succeed to relocate the object to.

To summarize, we introduced representations for our proposed data-driven pre-grasp strategy to readjust an object on a surface prior to grasping, which allows us to select preshapes $\mathbb{S}$ and pre-grasp manipulations $\mathbb{M}$ based on the object category $\mathbb{O}$. These data structures build the foundation for general usage of pre-grasp strategies to increase the success rate of stable grasp acquisition.

4.1.3. Preshape and pre-grasp manipulation for sliding

The introduced representation for pre-grasp strategies is capable of providing solutions for different kinds of pre-grasp manipulations. Here, we demonstrate the benefit based on sliding pre-grasp manipulation and the corresponding preshapes optimized for this task.

We informally observed in the video survey mentioned in Section 1 that preshapes for sliding pre-grasp manipulation have similar initial offsets to the object surface and similar distances to object edges. Hence, a set of starting poses P can be efficiently determined for each object within a certain category based on the following quantities in the object coordinate system, illustrated in Fig. 3:

- The initial position $\mathbf{x}_s$ on the object surface,
- the offset $\mathbf{f}$ of the hand to the object surface,
- the dimension $\mathbf{d}$ of the original object's bounding box,
- and the hand orientation stored by the roll axis $\mathbf{r}$ and yaw axis $\mathbf{y}$

The separation of the starting poses into the three parts, surface position, free-space offset, and orientation, ensure scale-invariant and rotation-invariant adaptation to objects of the same object category $\mathbb{O}$ as described in Section 4.3. The assigned pre-grasp manipulation $\mathbb{M}$ is always sliding manipulation. Constraints C in our context are fingertip contact with the object and collision free arm configuration for the pose. Every preshape has a set of grasp-types $\mathbf{G}$ which are reasonable based on the preshape knowledge, for example a large object is preferably grasped with two hands if it is reachable for both hands.

Sliding is the pre-grasp manipulation action a used in this paper. Constraints C for this $\mathbb{M}$ are that the fingers are in contact during the whole sliding manipulation, and continuous arm configurations for the whole path can be found. Object pose constraints are not considered due to the fact that no special grasps for objects such as handled ones are available.

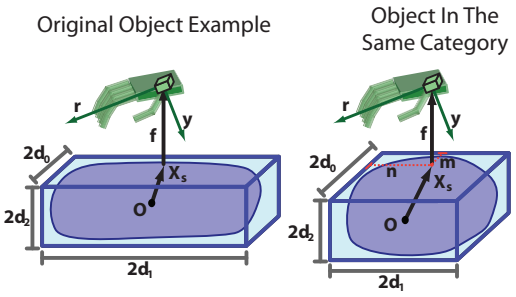


Figure 3: Representation of stored information in the preshape $\mathbb{S}$ structure. These components allow for automatic determination of the starting hand poses relative to the object. The left side shows the original object from the preshape example. The right side shows one possible starting pose made with absolute distance m and relative distance n for the two visible sides.

4.2. Architecture

Based on the presented data structures we propose a data driven framework which performs pre-grasp strategies to grasp an object located on a surface. The general procedure is visualized in Fig. 2. The method takes an object category $\mathbb{O}$ described in detail in Section 4.1 as input which can be gathered through a classification step to find the best matching $\mathbb{O}$ for a given object.

At first, the pre-grasp strategy computes the set of starting poses P for the preshapes affiliated with $\mathbb{O}$. This step enables starting pose computation that is invariant to object scaling and rotation as demonstrated by our sliding preshape representation.

In the second step kinematic template preshapes of $\mathbb{O}$ are adapted to the object surface. The *adapted preshapes* are *evaluated* with a rating function such as $Q_p(\mathbf{c}, \mathbf{p}, \mathbf{q}, F, C)$ to find the best adapted preshape in terms of Q_p which then is propagated to the pre-grasp manipulation phase. The rating function Q_p described in Section 5.2 checks finger contact with the object, the arm pose, the size of the final region F, and the compliance with constraints.

The next step performs a *corresponding pre-grasp manipulation* $\mathbb{M}$ to the best rated $\mathbb{S}$. A successful pre-grasp manipulation plan is achieved if the object is within the final region F satisfying the constraints C. If no successful plan is found the best manipulation regarding $Q_m(\mathbf{p}_f, F, C, g)$ is performed and the search is restarted at the previous step. Q_m described in Section 5.2 determines which manipulation will be performed based on the object pose, the distance to the final region, the observed constraints, and the grasp success.

Finally, the grasp-types $\mathbf{G}$ usable for a object pose $\mathbf{p}_f$ are *selected and adapted* to the object. If no successful solution respecting force closure metric can be found, the previous step is repeated to manipulate the object to another location. If no grasp solution can be found for a certain amount of trials another pre-grasp manipulation available by the adapted preshape is chosen to successful find a final grasp.

In the following Sections 4.3 to 4.6 we describe the four individual parts of the proposed pre-grasp strategy: start pose computation, pose adaptation, pre-grasp manipulation, and final grasping.

4.3. Preshape Starting Pose Computation

As described in Section 4.1 every preshape has to have the ability to serve a set of starting poses P. We present in this paper an efficient way to provide P for sliding pre-grasp manipulation based on the preshape optimized for sliding introduced in Section 4.1. In the remainder of this section we refer to this distinct preshape.

To regain the starting poses P we divided the storage into three parts. Only the first part, surface position $\mathbf{x}_s$ generates a set of points. For every surface point an offset $\mathbf{f}$ is added and the orientation of the hand is set based on the roll axis $\mathbf{r}$ and yaw axis $\mathbf{y}$. The latter two ensure that the hand orientation is independent of the object rotation using a right-handed coordinate convention, the orientation is additionally generated correctly regardless if it is a right or left hand. To generate starting poses regardless of the object pose we transform the object coordinate system so that the robot shoulder position is expressed by a positive vector.

In our implementation a preshape for sliding pre-grasp manipulation is always related to three object sides. We determine the three sides with respect to the new object coordinate system. There are two ways an example position can be retargeted to a new object's side: either the absolute or the relative distance to the side can be preserved. The absolute and relative distance either does not or does change with scaling, respectively. The absolute relative distance can be measured regarding to the positive or negative side of the coordinate system, thus there are always two possibilities for absolute distances. Hence, there is one starting position if the relation to all three sides is relative. If the relation to one side is measured absolute, there are 6 different solutions one shown in Fig. 3, for every side two distances. For two absolute distances there exist 12 possible solutions and if all sides distances are retargeted as absolute 8 possible starting positions are available. In total 27 starting positions $\mathbf{x}_s$ are necessary to express

all possible relations of the surface point with the object sides.

Thus, multiple surface positions are available, and this overhead is acceptable in order to store a set which contains a promising relative pose for a new object.

4.4. Preshape Adaptation and Evaluation

The second phase, preshape adaptation (Fig. 2), builds upon the set of preshapes selected by the object category and P determined by the previous step. The key aspect is that preshapes enable efficient computation of hand configurations suitable for pre-grasp manipulations considering environmental, robot, and object constraints.

Hence, the goal of this phase is to find suitable hand poses and configurations for the current object to perform related pre-grasp manipulations. Several solutions for this subtask are available in literature. For example Hsiao [17] proposed a solution for template grasp adaptation for direct grasping using starting position mapping on related objects. Kim [43] suggested a grasp adaption algorithm based on mapping grasp positions from an example to a new object. Another grasp adaptation method from generic prototypes is discussed by Pollard [16].

Since we need a grasp position for pre-grasp manipulation and not only for direct grasping, we present a new adaptation algorithm using additional context knowledge provided by the preshape structure. The data representation provides starting poses and corresponding hand configurations with a high probability for successful adaptation. The starting poses are checked for reachability to generate collision-free arm IK solutions. Hence, we do not initially search for starting preshapes, but we evaluate the preshapes which are reachable. To evaluate them, we need to adapt the preshapes to the current object. The adaptation process has two parts, one for adapting hand configurations – changing the hand configuration $\mathbf{c}$ to get fingertip contact, and another one to relocate the hand if no solution regarding Q_p can be determined (Fig. 4).

Now we describe the fingertip contact calculation. Due to the configuration $\mathbf{c}$ given by preshape $\mathbb{S}$, the hand is already in a promising configuration for contacts. Therefore, we individually search for finger contact with the object surface based on an iterative inverse kinematics approach for the current finger chain. This approach prevents awkward hand configurations because the initial finger configuration is used if no contact solution can be found. This method is described in detail in Fig. 4. Once the preshapes have been adapted to the object, we then compare them to select the best

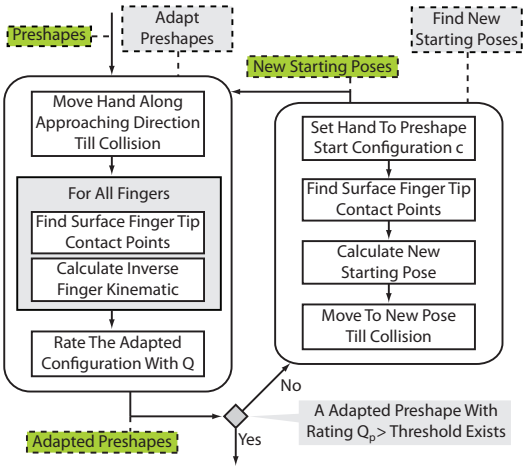


Figure 4: The left side of our proposed algorithm adapts a hand configuration of a given preshape to the object surface. If no adapted preshape with a high enough rating Q_p is available, the right side of the algorithm computes new starting poses.

adapted preshape for the pre-grasp manipulation phase. The adapted preshapes are scored by a rating function $Q_p(\mathbf{c}, \mathbf{p}, \mathbf{q}, F, C)$ regarding the pre-grasp manipulations they correspond to and the best is propagated to the pre-grasp manipulation phase.

Although the initial hand pose $\mathbf{p} \in P$ is likely to be in a promising spot for finding a successful solution with high value of Q_p, it may occur that no finger contacts can be found. If no preshape is successfully adapted, a new P has to be found to achieve successful solutions with respect to Q_p. In these cases, we use the original preshape configuration $\mathbf{c}$, and then we search for a new hand pose $\mathbf{p}$ based on the closest surface points to the active finger tips. After the hand is iteratively moved to the new pose, the process of finding fingertip contact restarts (Fig. 4).

4.5. Pre-grasp Manipulation

Currently, the pre-grasp strategy has achieved the following goals. A set of preshapes have been adapted to the object and the best adapted preshape relative to a quality function Q_p has been selected. Only pre-grasp manipulations related to the object category and adapted preshape are available at this stage. In addition to that the adapted preshape has to respect the constraints for the pre-grasp manipulation.

The goal of the pre-grasp manipulation phase is to find a path such that the object is relocated to the final region and respects the pose constraint of the selected

preshape, manipulation, and grasp. The kind of pre-grasp manipulation used for readjusting the object location is specified within the selected and adapted preshape $\mathbb{S}$ and the selected object category $\mathbb{O}$.

Pre-grasp strategies in general have two aspects in common. First, they use the benefit that the object rests on a surface, such that it is often it is easier to adjust the object pose than to grasp the object. Second, they solve the problem of finding successful and stable grasps if the object is not within a certain reachable area or pose by relocating the object to more easier plan lifting grasps.

The general idea of this paper to provide a representation that enables to use multiple pre-grasp manipulations such as sliding manipulations proposed by Lynch et al. [30], rotating manipulations discussed by Chang et al. [25], for a pre-grasp strategy to adjust the object prior to grasping.

With completion of this phase, the object is located within the final region and is satisfying the pose constraints, and thus the grasping action can be planned more easily.

4.6. Final Grasp

The final grasp is planned once the pre-grasp manipulation has successfully located the object within a final region F respecting constraints C. Finding grasp candidates for an object located in such a region has been well explored in robotics and computer graphics. Literature such as [13][20][21][16][17] about this topic was introduced in Section 2. If a successful grasp solution respecting force closure metric is found the pre-grasp strategy is finished. But if no stable solution can be found (Fig. 2), either the object has to be relocated based on the previous step, or the manipulation strategy has to be changed.

5. Implementation

We implemented our framework as a plugin in OpenRAVE [44] which provides collision checking, IK solutions and RRT planning to our simulation. The RRT motion planner is used to find a continuous motion between two distinct hand configurations in a certain pose.

For the sliding preshapes and pre-grasp manipulation the following assumptions are made. First, surface geometry models are available for all objects. Object weight is known a priori, and objects have an initial coordinate system with a known upright axis indicating how it rests on the surface. Additionally, our sliding implementation does not support obstacles on the surface. Other pre-grasp manipulation strategies as well as a more sophisticated sliding implementation are envisioned.

5.1. Object Classification

Mapping an input feature vector, e.g. weight and dimension of a given object, to an output label, e.g. box or cylinder, is a standard classification problem. There are many different solutions to solve this problem, for example neural networks [45] or support vector machines [46]. It is possible to use offline, online, supervised or unsupervised learning in our framework, which allows for flexible implementations.

In our implementation we use the dimensions, weight and curvature as input vector for a multilayer neural network. The dimensions and curvature are computed online based on the object trimesh. This classification results in assigning an object category. The classifier was trained offline with two manually selected object examples for every object category.

The goal of the object classification is to select the best matching object category for a given object so that the proposed pre-grasp strategy can prepare the object pose for final grasping. Also note that our method will not necessarily fail in response to misclassification of objects, due to the adaptation process that is described in Section 4.4. The object category is described in detail in Section 4.1.

5.2. Rating Function

As introduced in Section 4.4 the adapted preshapes are evaluated by the corresponding rating function $Q_p(\mathbf{c}, \mathbf{p}, \mathbf{q}, F, C)$. We propose a rating function which prefers more finger contacts, a large final region size, as many pre-grasp manipulations as possible, and unconstrained arm solutions regarding joint limits:

$$Q_p(\mathbf{c}, \mathbf{p}, \mathbf{q}, F, C) = \alpha_1 \sum_{i=1}^{n} b_i + \alpha_2 \, size(F) + \alpha_3 \, sat(C) + \alpha_4 \sum_{i=1}^{a} \left(\frac{(\min_i - q_i)^2 + (\max_x - q_i)^2}{(\min_i - \max_i)^2} \right)^{-1} \tag{4}$$

where n is the number of fingers, a the number of arm joints, b_i is a binary indicator expressing whether the finger i made contact. The relative size of the final region F is determined by $size(F)$. The constraints C are checked to determine if a related pre-grasp manipulation can be performed, for example if the force on the object is sufficiently away from the limit force. The joint limits are $\min_i$ and $\max_i$, the current

joint value is q_i. The different parts of the equation are weighted by α_i.

A second rating function Q_m introduced in Section 4.5 is designed to determine the pose $\mathbf{p}_f$ for unsuccessful pre-grasp manipulation which is the best starting pose for a new adaptation set.

$$Q_m(\mathbf{p}_f, F, C, g) = \alpha_1\, d(\mathbf{p}_f, F) + \alpha_2\, d(\mathbf{p}_f, C) + \alpha_2\, g \quad (5)$$

where the distance to the final region F is measured as well as to object pose constraints. In addition if the object was within the final region and not graspable determined by g, the final region for this grasp-type has to be removed for later trials.

5.3. Pre-grasp Manipulation

As mentioned in Section 4.1 sliding is the only pre-grasp manipulation implemented for evaluation in this paper. This is based on the observation within the video survey mentioned in Section 1 where sliding manipulation has been the most common pre-grasp manipulation strategy for the given objects. For other pre-grasp strategies like pre-grasp rotation we refer to the work of Chang et al. [25, 27, 40]; pre-grasp toppling and tumbling to Lynch at al. [32]; sliding by Dogar et al. [47].

For a successful sliding manipulation we try to find a continuous trajectory from the initial adapted preshape pose to a manipulation end pose so that the object is within the final region without losing the hand/object and the object/surface contacts. For sliding manipulation we have no pose constraint within the final region due to the fact that the manipulating hand is already in contact with the object during the whole manipulation. Due to the definition of the final region (Section 4.1) it is assumed that after successful pre-grasp manipulation the object is more likely to be graspable after successful pre-grasp manipulation.

When the hand is in contact with the object during sliding pre-grasp manipulation, we require that the finger object friction is high enough such that the object moves along with the hand movement on the surface, which is assumed to be true as long as contact between the finger and the object is maintained.

5.4. Final Grasp

The benefit in our framework is that object knowledge is already available through selected template grasps and, more importantly, the object is in a better location for grasping. Berenson et al. [20] propose a pre-computation of possible grasp candidates sampling the object surface to gather starting positions to adapt template grasp. Due to the better location and provided template grasps based on object knowledge given by the selected preshape $\mathbb{S}$ we sample the object surface and adapt the provided template grasps to the object surface with the algorithm described in Fig. 4.

After grasp candidates are found the stability has to be determined. A stable solution in this context has to satisfy the force-closure metric proposed by Ferrari et al. [23].

Table 1: Objects and object categories*.

$\mathbb{O}_0$ flat shape	$\mathbb{O}_1$ lightweight box	$\mathbb{O}_2$ heavy box	$\mathbb{O}_3$ cylinder
CD	baseball bat	dictionary	cookie tin
credit card	stapler	keyboard box	sugar canister
15-cm ruler	tape dispenser	VGA-splitter	plant pot
house key	cassette tape	food storage container/box	water jug
	bed linen box		
	portable hard-drive		

*All objects were manipulated in the human study. Objects in the first row, and other objects, were tested in the simulation validation. The objects in the second and third rows provided the examples for preshape and manipulation information in the database.

6. Experiments and Results

We compared our method in simulation to a traditional planner for direct object grasping. We evaluated the success rate of finding a feasible object acquisition plan, and the computation time.

6.1. Scenario

In the test scenario, there is a single table in front of the robot. The robot is a bi-manual humanoid model from OpenRAVE [44] with 7-DoF arms and 15-DoF hands (Fig. 1). In our implementation, the right hand has been replaced with the ShadowHand with 23 DoFs.

Two objects from each of the four sliding manipulation preshape sets were tested (Table 1). Each preshape set consists of five preshapes from two objects in the same category, which were manually extracted from human examples gained with a motion capture experiment. The tested objects were not included in the database examples. For the final grasps we provide 12 grasp-types to both planners. Our method selects suitable grasp-types corresponding to the selected and adapted preshape. The direct grasp planner randomly selects one out of the 12 grasp-types and tries to find a stable lifting grasp, until a solution is found or all 12 are tried. We limited the search for a stable solution for each grasp-type to 20s.

Each object is placed at a random position and plane orientation on the table within a 1.2m×0.6m region in

front of the robot, shifted 0.4 m right of the robot center. We discard the random position if the object is not within the reaching radius of the robot's right arm. We generate poses in this manner to obtain 15 reachable poses per object. Both planners attempt an object acquisition solution for the same 15 starting object poses.

6.2. Direct Grasp Planner

For direct grasp planning the implementation described in Section 4.6 was used but because no object context is available, grasp-types available for the robot are randomly selected and adapted.

6.3. Simulation Results

Table 2 presents the results, separated for each object category and planning phase. The "Pre-grasp" column for our approach includes the object classification, calculation of the preshape starting poses, preshape adaptation and evaluation, and pre-grasp manipulation. The "Grasp" column contains the final grasp adaptation for both our method and the direct grasp planner. The "Trajectory" column consists of the trajectory plan from the initial position to the adapted final grasp for both methods: direct grasp planning, as well as for our approach if the object is initially in the final region F of the preshape. If not, in our approach the trajectory consists of two separate trajectories, one from the initial pose to the adapted sliding preshape and a second one from final sliding to the final grasp pose.

Our strategy increases the success rate for object acquisition for all tests as shown in Table 2. In addition, our approach reduces the computation time for grasp adaptation significantly regardless of whether the object is directly graspable or not. Furthermore median of the planning time for the grasp pose with our method was about 2s whereas the direct grasp planned took about 200s. The long computation time for the direct grasp planner results through the exhaustive and in the most cases not successful search for stable final grasp candidates.

Figure 5 shows example simulation results for different object categories.

6.4. Perceptual Evaluation

We also evaluated human response to the pre-grasp manipulation plans and direct grasping plans. In our survey, 21 participants viewed pairs of simulation videos showing the humanoid agent using, in a random order, either pre-grasp manipulation or direct grasping. Participants selected the preferred video in each pair. Table 3 shows that pre-grasp rotation was preferred by more people for the cookie tin, baseball bat, and linen box objects. A chi-square test on the number of participants preferring pre-grasp manipulation or direct grasping for at least 3 of the 5 video pairs rejected that the ratio was balanced 50-50 ($p(X^2 = 5.76, df = 1) = 0.02$).

Table 2: Simulation results for the method comparison.

	Successes	Mean planning times (seconds)		
	out of 36	Pre-grasp	Grasp	Trajectory
$\mathbb{O}_0$: CD, ruler				
Pre-grasp push	36	1.5	2.0	15.0
Direct grasp	1	–	53.4	0.4
$\mathbb{O}_1$: bat, stapler				
Pre-grasp push	34	3.4	15.2	16.7
Direct grasp	25	–	15.8	11.8
$\mathbb{O}_2$: book, food box				
Pre-grasp push	36	1.5	3.2	17.7
Direct grasp	26	–	16.5	8.4
$\mathbb{O}_3$: tin, jug				
Pre-grasp push	36	2.1	2.6	18.6
Direct grasp	25	–	33.6	8.9
Total	out of 144	Pre-grasp	Grasp	Trajectory
Pre-grasp push	142	2.1	5.6	17.0
Direct grasp	77	–	22.3	9.6

Table 3: The number of participants who preferred either pre-grasp interaction or direct grasping in the video survey.

	Manipulation method	
Object	Direct grasp	Pre-grasp push
Cookie tin	3	18
Dictionary	10	11
Baseball bat	6	15
CD	9	12
Linen box	5	16
Overall preference ($\geq 3/5$ objects)	5	16

6.5. Physical Demonstration

We demonstrated the physical plausibility of our simulated pre-grasp strategy plans on a multi-fingered robot manipulator. The system consists of a 7-DoF Motoman arm and an attached Shadowhand robot with 5 fingers. In our example demonstration, the object is a CD, which is is difficult to grasp from a table because of its thin edge. However, the Motoman with Shadowhand was able to grasp the CD after first using a sliding pre-grasp manipulation planned with our method.

The CD was manually placed on the table to match the simulated task scene. The Motoman arm trajectory produced by our simulation method was executed open-loop on the robot. Due to limitations of the

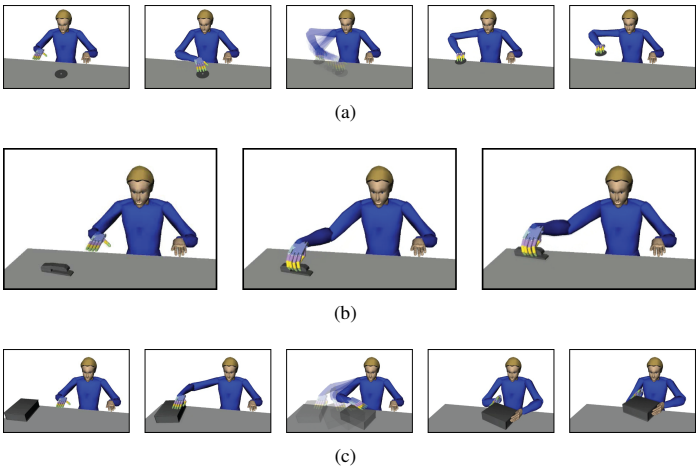


Figure 5: Example simulation results.

control synchronization, the hand preshapes for the Shadowhand were selected from our simulated plan but were manually pre-set to match the arm trajectory timing. The video at `http://his.anthropomatik.kit.edu/english/532.php` shows the comparison between the simulated Motoman plan and the physical execution for the CD object.

7. Validation of template classification

The experiments in the previous section demonstrate the utility of augmenting the grasping process with pre-grasp interaction for object acquisition. Using our framework for planning pre-grasp sliding interactions, we were able to synthesize plausible and natural-looking actions that improved the reachability of the object for grasping.

We now examine how both the initial template classification and the later template adaptation are critical for finding pre-grasp interaction plans within a tractable time. To validate these stages, we consider experiments with two modified versions of the method presented in Section 4.

First, we test the value of organizing examples according to the object context. We modify our framework by eliminating object classification while retaining the information from the same examples of pre-grasp hand shapes and poses relative to the objects. In essence, this reduces the classification step to predicting a single class that includes all of the example data. Thus we still use the examples to find promising hand pre-shapes, but the selection is agnostic to the object features.

Second, we test the robustness to misclassification of the object class. In this set of experiments, the examples are organized according to the same four object classes described in Table 1. However, we deliberately misassign the classification result in order to investigate the response of the hand adaptation process.

7.1. Example organization in a single-class

In this experiment, we keep our original framework but instead input a database consisting of a single class which includes all example hand preshapes. This is equivalent to omitting the classification of the object features into an object category to determine a subset of the examples. Thus the candidates for initial hand poses and preshapes are selected by evaluating and attempting adaptaption with all examples in the database in a random order. The final grasp shape and final region for object location are selected in the same manner as before, that is according to the corresponding preshape that is attempted during the pre-grasp pushing manipulation.

Due to the increase in the number of examples to test, an additional termination criterion is included to constrain the length of the experiment. The planning process was terminated at a total time of 15 minutes if no successful grasp for object acquisition is found. This prevents the experiment from testing every example to exhaustion. Only the pre-grasp interaction strategy was tested, without the direct grasping comparison.

The results in Table 4 indicate that the lack of organizing examples by object class decreased the success of finding feasible grasping plans by 9% compared to the

Table 4: Simulation results using a single class template.

	Successes	Mean planning times (seconds)		
	out of 36	Pre-grasp	Grasp	Trajectory
$\mathbb{O}_0$: CD, ruler Pre-grasp push	27	75.7	124.0	11.6
$\mathbb{O}_1$: bat, stapler Pre-grasp push	36	16.0	10.6	189.7
$\mathbb{O}_2$: book, food box Pre-grasp push	31	3.2	83.0	17.8
$\mathbb{O}_3$: tin, jug Pre-grasp push	35	5.8	25.3	22.4
Total	out of 144	Pre-grasp	Grasp	Trajectory
Pre-grasp push	129	22.7	55.7	65.7

original framework results in Table 2. In addition, the average computational time required for planning the manipulation action increased by 11-fold, 10-fold, and 4-fold for the pre-grasp interaction pushing, the final grasp planning, and the arm trajectory planning. These results show the benefit of having examples organized in a manner that allows the identification of promising candidate configurations.

7.2. Results for object category misclassification

In this second validation experiment, we test the adaptation phase of our framework.

The previous validation experiment demonstrated the need for example organization into template categories. In our original experiments presented in Section 6, the classification results appeared reasonable even for objects that did not necessarily fit the semantic labels we used in Table 1. For example, the water jug object has a square cross section in the bottom half of its the base, but it was classified in the "cylinder" category $\mathbb{O}_3$ not the "heavy box" category $\mathbb{O}_2$. This resulted in natural-looking pre-grasp interaction where the hand contact the side surfaces of the jug similar to the lateral surfaces of a cylinder, instead of interaction with hand contact at the edge between the top and side faces of a box.

For additional novel objects or a different object classifier, however, it is possible that the classification method may result in a low confidence between two or more classes that would be similarly appropriate for the object. Here we test how the adaptation phase in our framework can be used to robustly modify the hand preshapes to new objects when the classification results are different.

In this experiment, our framework remains unchanged except the output of the classification result. We force the classification result to swap between two pairs of categories:

- if the original classification result would have been $\mathbf{O}_0$ for thin objects, the mis-classified result is output as $\mathbf{O}_1$ for light weight boxes, and vice versa.
- if the original classification result would have been $\mathbf{O}_2$ for heavy boxes, the mis-classified result is output as $\mathbf{O}_3$ for cylinders, and vice versa.

We did not consider the extreme mis-classification results of swapping, e.g., $\mathbf{O}_0$ for thin objects with $\mathbf{O}_2$ for heavy boxes, since this result is unlikely if the classifier has been trained on sufficient examples.

The results in Table 5 for the misclassification tests indicate that there was a decrease in the success rate for finding feasible grasp plans, compared to the "correct" classification results in Table 2 (125 successful plans overall instead of 142). Interestingly, the direct grasp successes increased for the mis-classification results. This was due to the fact that the final region for the object grasping is associated with the mis-classified object class. Thus, sometimes a partial pushing interaction is initiated that moves the object from the starting location. For direct grasping, there is no object motion from the start location, and the misclassification in some examples resulted in different hand preshapes being used for successful grasping.

The comparison for the timing results (Table 5) with the original computation time averages (Table 2) indicates that the main increase in planning time occurs in the grasp planning phase, along with an increase in the pre-grasp push planning time. This is due to the association of the final grasp hand shapes with a selected pushing preshape. This association leads to time-efficient selection and natural-looking manipulation actions when the object is "correctly" classified. However, in the mis-classification case, the associated final grasp shapes would not be appropriate for the object. This point in fact demonstrates the utility of using pre-grasp interactions because the hand shapes for pushing or other pre-grasp interactions are usually less restricted than the final grasps required for lifting an object. That is, the adaptation of a pushing shape primarily needs to make contact with the object to exert forces in the right direction, but the grasping shape must satisfy more constraints to be a feasible lifting grasp. In systems where the classification confidence is low, pre-grasp template adaptation using the selected class may still be reasonable, but the final grasping phase could be altered to consider hand shapes from similar grasp classes instead of only the selected class.

Table 5: Simulation results for mis-classified template examples. Note the changes in object class label $\mathbf{O}_i$ compared to Table 2.

	Successes	Mean planning times (seconds)		
	out of 36	Pre-grasp	Grasp	Trajectory
$\mathbb{O}_1$: CD, ruler				
Pre-grasp push	22	7.4	220.8	8.8
Direct grasp	1	–	111.3	0.4
$\mathbb{O}_0$: bat, stapler				
Pre-grasp push	31	16.6	12.2	18.2
Direct grasp	30	–	12.5	8.8
$\mathbb{O}_3$: book, food box				
Pre-grasp push	36	3.3	2.8	17.2
Direct grasp	28	–	22.7	8.8
$\mathbb{O}_2$: tin, jug				
Pre-grasp push	36	2.1	2.6	19.7
Direct grasp	24	–	30.0	8.9
Total	out of 144	Pre-grasp	Grasp	Trajectory
Pre-grasp push	125	7.0	43.4	16.7
Direct grasp	83	–	22.2	8.7

8. Discussion

In this paper we presented a framework for representing and re-synthesizing examples of pre-grasp interaction, particularly sliding actions, that can improve grasping success. Our approach was based on patterns observed in human demonstration of pre-grasp interaction, where the choice of hand position relative to the object and hand shape were similar for similar object features. These examples simplify the high-dimensional search for candidate hand configurations but providing promising templates for a new object based on its object category. With this reduction of candidate configurations, the search becomes tractable for articulated manipulators with multi-fingered hands.

The basis of our framework is the representation of a pre-grasp interaction that includes context information including the hand preshape and pose as well as other constraints such as the final region suitable for the final grasp. Another key component of our method is the representation of the candidate preshape poses relative to the object, which are stored and regenerated to account for changes in object scaling and rotation. Altogether, these pre-grasp manipulation actions, which includes all kinds of prior adjustments to object acquisition, and final grasping can be computed online. The whole strategy results in more robust and stable object grasping.

8.1. Organization of examples

Our validation of the proposed framework focused on the importance of organizing examples for efficient reuse. In particular, when there are several examples of candidate hand preshapes, it is not sufficient that the examples exist in the reference database. Instead, organization — in this case by object categories – allowed our method to quickly determine a subset of examples that were suitable for the simulation tasks. The efficiency of identifying promising templates is an aspect of example-based planning that will be even more critical for larger databases of manipulation actions.

In our current implementation, the database of preshape examples was manually organized based on our observation from the human motion capture studies. We found that examples from a small set of 8 objects and 4 participants provided a promising templates that could be adapted successfully to new objects. An extension that is beyond the scope of the current work is to extend the system to include automatic learning or updates of the object classes and examples. This can be done with both new human-demonstrated examples as well as examples that come directly from the robot's manipulation that result from our syntheis method.

8.2. Directions for future extensions

In future work, we plan to extend our database and representations to accommodate additional types of pre-grasp manipulation such as tumbling or topping. While other action modes and preshape data structure to support the thesis and to update the sliding one to a more sophisticated version which is capable of avoiding obstacles on the surface.

Future steps may explore reducing the assumptions about the friction coefficients in the environment. Initially planning with friction assumptions, but then using feedback during the pre-grasp manipulation to obtain new object parameters such as inertia and friction coefficients would increase the stability of the final grasp due to the extra knowledge gained.

Another interesting idea is to parallelize different planning steps. As soon as the object representation is selected, all possible preshapes are available. Hence, one approach is to evaluate final grasp poses in parallel which is done in [20] by Berenson et al. as offline precomputation. Preshape adaptation can also be parallelized due to orthogonal usage. This would speed up final grasp planning because it is only a selection of possible grasp solutions with respect to environmental restrictions.

Overall, the proposed unified representation of pre-grasp strategies for object manipulation significantly increases the object acquisition success rate. This underlines the great potential of using human behavior knowledge to develop new planning strategies.

References

[1] T. Asfour, K. Regenstein, P. Azad, J. Schroder, A. Bierbaum, N. Vahrenkamp, R. Dillmann, ARMAR-III: An integrated humanoid platform for sensory-motor control, in: Proc. IEEE Int. Conf. Humanoid Robots (Humanoids), 2006, pp. 169–175.

[2] T. Wimbock, C. Ott, G. Hirzinger, Impedance Behaviors for Two-handed Manipulation: Design and Experiments, Proc. IEEE Int. Conf. Robots Automation (ICRA) (2007) 4182–4189.

[3] K. Okada, T. Ogura, A. Haneda, J. Fujimoto, F. Gravot, M. Inaba, Humanoid motion generation system on HRP2-JSK for daily life environment, IEEE International Conference Mechatronics and Automation (2005) 1772–1777.

[4] M. R. Cutkosky, On grasp choice, grasp models, and the design of hands for manufacturing tasks 5 (3) (1989) 269–279.

[5] J. R. Napier, Hands, Princeton University Press, Princeton, New Jersey, 1993.

[6] S. J. Edwards, D. J. Buckland, J. D. McCoy-Powlen, Developmental & Functional Hand Grasps, Slack Incorporated, Thorofare, New Jersey, 2002.

[7] K. Ikeuchi, T. Suehiro, Toward an assembly plan from observation Part I. Task recognition with polyhedral objects 10 (3) (1994) 368–385. doi:10.1109/70.294211.

[8] S. B. Kang, K. Ikeuchi, Toward automatic robot instruction from perception-mapping human grasps to manipulator grasps 13 (1) (1997) 81–95. doi:10.1109/70.554349.

[9] S. Ekvall, D. Kragic, Grasp recognition for programming by demonstration, 2005, pp. 748–753.

[10] K. Bernardin, K. Ogawara, K. Ikeuchi, R. Dillmann, A sensor fusion approach for recognizing continuous human grasping sequences using hidden Markov models 21 (1) (2005) 47–57.

[11] H. Kjellstrom, J. Romero, D. Kragic, Visual recognition of grasps for human-to-robot mapping, in: Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on, 2008, pp. 3192 –3199. doi:10.1109/IROS.2008.4650917.

[12] S. El-Khoury, A. Sahbani, V. Perdereau, Learning the natural grasping component of an unknown object, in: Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on, 2007, pp. 2957 –2962. doi:10.1109/IROS.2007.4399052.

[13] A. Miller, S. Knoop, H. Christensen, P. Allen, Automatic grasp planning using shape primitives, in: Proc. IEEE Int. Conf. Robots Automation (ICRA), Vol. 2, 2003, pp. 1824–1829.

[14] C. Goldfeder, P. Allen, C. Lackner, R. Pelossof, Grasp planning via decomposition trees, in: Proc. IEEE Int. Conf. Robots Automation (ICRA), 2007, pp. 10–14.

[15] Y. Li, N. Pollard, A shape matching algorithm for synthesizing humanlike enveloping grasps, in: IEEE-RAS International Conference on Humanoid Robots (Humanoids 2005), 2005.

[16] N. Pollard, Synthesizing grasps from generalized prototypes, in: Proc. IEEE Int. Conf. Robots Automation (ICRA), 1996, pp. 2124–2130.

[17] K. Hsiao, T. Lozano-Perez, Imitation learning of whole-body grasps, in: Proc. IEEE Conf. Intelligent Robots and Systems (IROS), 2006, pp. 5657 –5662.

[18] A. Saxena, J. Driemeyer, J. Kearns, A. Ng, Robotic grasping of novel objects, Advances in Neural Information Processing Systems (NIPS).

[19] M. Ciocarlie, C. Goldfeder, P. Allen, Dimensionality reduction for hand-independent dexterous robotic grasping, in: IROS, San Diego, CA, 2007.

[20] D. Berenson, R. Diankov, K. Nishiwaki, S. Kagami, J. Kuffner, Grasp planning in complex scenes, in: Proc. IEEE Int. Conf. Humanoid Robots (Humanoids), 2007, pp. 42–48.

[21] M. Przybylski, T. Asfour, R. Dillmann, Unions of balls for shape approximation in robot grasping, in: To appear in: Proc. IEEE Conf. Intelligent Robots and Systems (IROS), 2010.

[22] A. Bicchi, V. Kumar, Robotic grasping and contact: a review, in: Proc. IEEE Int. Conf. Robots Automation (ICRA), 2000, pp. 348–353.

[23] C. Ferrari, J. Canny, Planning optimal grasps, in: Proc. IEEE Int. Conf. Robots Automation (ICRA), 1992, pp. 2290–2295.

[24] A. T. Miller, P. K. Allen, Examples of 3D Grasp Quality Computations, Proc. IEEE Int. Conf. Robots Automation (ICRA) 2.

[25] L. Y. Chang, G. Zeglin, N. Pollard, Preparatory object rotation as a human-inspired grasping strategy, in: Proc. IEEE Int. Conf. Humanoid Robots (Humanoids), 2008, pp. 527–534.

[26] K. Hsiao, L. Kaelbling, T. Lozano-Pérez, Task-Driven Tactile Exploration.

[27] L. Y. Chang, S. Srinivasa, N. Pollard, Planning pre-grasp manipulation for transport tasks, in: Proc. IEEE Int. Conf. Robots Automation (ICRA), 2010.

[28] M. T. Mason, Mechanics of Robotic Manipulation, Cambridge, MA: MIT Press.

[29] T. Lozano-Perez, M. Mason, R. Taylor, Automatic synthesis of fine-motion strategies for robots, The International Journal of Robotics Research (1984) 13–24.

[30] K. Lynch, M. Mason, Stable pushing: Mechanics, controllability, and planning, The International Journal of Robotics Research.

[31] M. Dogar, S. Srinivasa, Push-grasping with dexterous hands: Mechanics and a method, in: Proceedings of 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2010), 2010.

[32] K. Lynch, Toppling manipulation, in: Proc. IEEE Int. Conf. Robots Automation (ICRA), 1999, pp. 2551–2557.

[33] K. Hauser, V. Ng-Thow-Hing, Randomized multi-modal motion planning for a humanoid robot manipulation task, The International Journal of Robotics Research-doi:10.1177/0278364910386985.

[34] E. Yoshida, M. Poirier, J.-P. Laumond, O. Kanoun, F. Lamiraux, R. Alami, K. Yokoi, Pivoting based manipulation by a humanoid robot, Autonomous Robots 28 (1) (2010) 77–88. doi:10.1007/s10514-009-9143-x.

[35] D. Bertram, J. Kuffner, R. Dillmann, T. Asfour, An integrated approach to inverse kinematics and path planning for redundant manipulators, Proc. IEEE Int. Conf. Robots Automation (ICRA) (2006) 1874–1879.

[36] S. R. Buss, Introduction to inverse kinematics with Jacobian transpose, pseudoinverse and damped least squares methods., ucsd.edu/˜sbuss/ResearchWeb.

[37] D. Tolani, A. Goswami, N. Badler, Real-time inverse kinematics techniques for anthropomorphic limbs, Graphical models (2000) 353–388.

[38] J. Latombe, Robot motion planning, Springer, 1991.

[39] S. LaValle, Rapidly-exploring random trees: A new tool for path planning, Tech. rep., TR 98-11, Computer Science Dept., Iowa State University (1998).

[40] L. Y. Chang, N. S. Pollard, Video survey of pre-grasp interactions in natural hand activities, in: Robotics: Science and Systems (RSS) 2009 Workshop: Understanding the Human Hand for Advancing Robotic Manipulation, University of Washington, Seattle, USA, 2009.
URL `http://www.cs.washington.edu/homes/bravi/rssws/paper-15-chang.pdf`

[41] L. Y. Chang, R. L. Klatzky, N. S. Pollard, Selection criteria for preparatory object rotation in manual lifting actions, J. Motor Behavior 42 (1) (2010) 11–27.

[42] L. Y. Chang, G. J. Zeglin, N. S. Pollard, Preparatory object rotation as a human-inspired grasping strategy, 2008, pp. 527–534.

[43] J. Kim, Example-based grasp adaptation, Master's thesis, Dept. of Electrical Engineering and Computer Science, Massachusetts Institute of Technology (2007).
[44] R. Diankov, J. Kuffner, Openrave: A planning architecture for autonomous robotics, Tech. Rep. CMU-RI-TR-08-34, Robotics Institute (2008).
[45] S. Haykin, Neural networks: a comprehensive foundation, Prentice Hall, 2008.
[46] C. Burges, A tutorial on support vector machines for pattern recognition, Data mining and knowledge discovery (1998) 121–167.
[47] M. Dogar, S. Srinivasa, Push-grasping with dexterous hands: Mechanics and a method, in: Proc. IEEE Conf. Intelligent Robots and Systems (IROS), 2010.

***Biography of each author**

Daniel Kappler is currently a Diploma student in Computer Science at Karlsruhe Institute of Technology, Karlsruhe, Germany.
His research interests include humanoid robotics, dexterous manipulation, grasping and biological inspired machine learning.

Lillian Chang is a postdoctoral research fellow with Intel Labs Seattle and a Research Associate at the University of Washington. She completed her Ph.D. in Robotics at the Robotics Institute in the School of Computer Science at Carnegie Mellon University. Her research interests include dexterous manipulation, motor control in both human and robotic systems, and human-robot interaction in physical manipulation tasks.

Lillian is a 2010 Computing Innovation (CI) Fellow, awarded by the Computing Research Association and the Computrng Community Consortium with support from the National Science Foundation. She is also a former NASA Harriet G. Jenkins Pre-doctoral Fellow and a former NSF Graduate Research Fellow.

Nancy Pollard is an Associate Professor in the Robotics Institute and the Computer Science Department at Carnegie Mellon University. She received her PhD in Electrical Engineering and Computer Science from the MIT Artificial Intelligence Laboratory in 1994, where she performed research on grasp planning for articulated robot hands. Before joining CMU, Nancy was an Assistant Professor and part of the Computer Graphics Group at Brown University. Her primary research objective is to understand how to create natural motion for animated human characters and humanoid robots.

Tamim Asfour is senior research scientist and leader of the Humanoid Research Group at Humanoids and Intelligence Systems Lab, Institute for Anthropomatics, Karlsruhe Institute of Technology (KIT). He received his diploma degree in Electrical Engineering (Dipl.-Ing.) in 1994 and his PhD in Computer Science (Dr.-Ing.) in 2003 from the University of Karlsruhe. In 2003 he was awarded with the Research Center for Information Technology (FZI) price for his outstanding Ph.D. thesis on sensorimotor control in humanoid robotics and the development of the humanoid robot ARMAR. Since September 2010 he holds an Adjunct Professor position at the Georgia Institute of Technology (Georgia Tech), College of Computing, School of Interactive Computing.

Tamim Asfour is member of the Editorial Board of IEEE Transactions on Robotics and European Chair of the IEEE-RAS Technical Committee on Humanoid Robots. He is member the Executive Board of the German Association of Robotics (DGR: Deutsche Gesellschaft für Robotik). He serves as member on several program committees and review panels. He is leading the system integration tasks and the development team of the humanoid robot series ARMAR in the German Humanoid Robotics Project (SFB 588) funded by the German Research Foundation (DFG). He is Principle Investigator in the Cognitive Systems projects Xperience, GRASP and PACO-PLUS, funded by the European Commission.

His major research interest is humanoid robotics. In particular, his research topics include action learning from human observation, goal-directed imitation learning, dexterous grasping and manipulation, active vision and active touch, whole-body motion planning, cognitive control architectures, system integration, robot software and hardware control architecture, motor control and mechatronics.

R\"{u}diger Dillmann is full professor at the Karlsruhe Institute of Technology (KIT), Department of Informatics, and head of the Humanoids and Intelligence Systems Laboratory at the Institute for Anthropomatics. He is member of the Executive Board of the Technology Research Transfer Center (FZI). He received his PhD in Computer Science in 1980 and his habilitation on "Learning Robots" in 1986 from the University of Karlsruhe. In 1981-1986 he held an associate professor position at the Institute for Process Control and Robotics (IPR), Karlsruhe. Since 1987 he is full professor at the Department of Informatics at KIT. He is the scientific leader and coordinator of the German collaborative research centre Humanoid Robots (SFB 588), coordinator of the integrated European Cognitive Systems projects PACO-PLUS and Xperience funded by the European Commission.

R\"{u}diger Dillmann is member of numerous international and national associations such as IEEE/RAS, VDI-GMA. He was member advisory committee of the IEEE Robotics and Automation Society (IEEE-RAS) and general and program chair of several international robotics conferences and member of the technical program committees of numerous international conferences. He is editor in chief for the International Journal Robotics and Autonomous Systems, editor in chief of the Springer book series "Cognitive Systems Monographs (COSMOS)" and member of the editorial advisory board of the Springer book series "Tracts in Advanced Robotics (STAR)"

His major research interests include humanoid robotics and human-centered robotics, learning from human observation and robot programming by demonstration, machine learning, active vision for human motion capture and scene understanding, cognitive cars, service robots, and medical applications of informatics.

***Photo of each author**

Example video

Click here to download Supplementary Material for on-line publication only: robots.wmv

Towards a Unifying Grasp Representation for Imitation Learning on Humanoid Robots

Martin Do, Tamim Asfour and Rüdiger Dillmann

Abstract— **In this paper, we present a grasp representation in task space exploiting position information of the fingertips. We propose a new way for grasp representation in the task space, which provides a suitable basis for grasp imitation learning. Inspired by neuroscientific findings, finger movement synergies in the task space together with fingertip positions are used to derive a parametric low-dimensional grasp representation. Taking into account correlating finger movements, we describe grasps using a system of virtual springs to connect the fingers, where different grasp types are defined by parameterizing the spring constants. Based on such continuous parameterization, all instantiation of grasp types and all hand preshapes during a grasping action (reach, preshape, enclose, open) can be represented. We present experimental results, in which the spring constants are merely estimated from fingertip motion tracking using a stereo camera setup of a humanoid robot. The results show that the generated grasps based on the proposed representation are similar to the observed grasps.**

I. Introduction

The acquisition of novel grasping skills plays an essential role in enabling humanoid robots to fully interact with the environment and the human. Considering the variety of objects and the different ways that an object can be dealt with, grasping strategies have to be developed which go beyond simple closure grasps towards a complete projection of the numerous possible grasps associated with each object. In order to bootstrap this process, imitation learning provides a fast alternative in terms of acquisition of new skills. Learning from human demonstration features the possibility of generating a representation of a demonstrated action which encodes human-likeness and subtle characteristics such as constraints which are satisfied during the execution of a specific task by a human.

Therefore, an essential issue in imitation learning that has to be addressed, is the question of what features have to be stored and processed and how, in order to obtain a generalized representation, which can be adapted and applied to new objects and situations. This issue becomes even more evident, when we look at the grasp problem for a robot hand where the motion of a highly complex system with several degrees of freedom (DoF) has to be controlled.

In [1], an early attempt of a generalized grasp representation in joint space is given by the concept of the grasp taxonomy which describes the assignment of grasp hand postures to a finite number of classes. Based on this grasp taxonomy [2] provides an extension which additionally incorporates a number of hand posture patterns common in a manufacturing environment. This concept suggests that given a class, represented by an exemplary posture, for a specific object, an instance in the vicinity of the posture can be found which forms a good grasp. In previous works such as [3] and [4] grasp taxonomies are applied to grasp synthesis. However, due to large number of DoF being involved, the configuration space remains huge.

Neuroscientific studies (see [5]) demonstrated that a lesser number of DoF needs to be actively controlled to cover the range of possible human hand postures during daily grasping activities, due to consistent covariations between the finger joints, known as postural hand synergies. In previous works ([6],[7],[8]), this concept has been applied to control the entire hand by a lower dimensional set of base postures commonly extracted by applying dimensionality reduction algorithms such as Principal Component Analysis. A similar approach is proposed in [9] where a grasp representation is generated in the form of manipulation manifolds consisting of hand postures and a mapping from joint space onto manipulation parameter in task space.

Nevertheless, these approaches operate in joint angle space, respectively in its projection to a lower dimensional subspace, which features unfavorable characteristics in terms of imitation learning through the observation by a humanoid robot. One major issue lies in the high complexity of observing and tracking human hands in joint space. Vision-based tracking algorithms which can be used with a stereo camera setup of a humanoid head do not provide the necessary performance and accuracy, while highly accurate motion capture systems involve high costs and time-consuming operation. Furthermore, the question arises how and whether the continuous reach movement in task space can be properly aligned with discrete hand postures in joint space. As stated in [10] and [11], the preshape and the enclose phase, respectively the final grasp phase, are accompanied by the reaching movement. Therefore, from the kinematic point of view, it seems to be reasonable to treat the whole grasping process as a single unit in which the three phases persist in permanent correlation to each other. Hence, attaining human-likeness is contradictory to the decoupled processing of the preshape, reach, and enclose phase. Ordinary task space representations (see [12], [13]) considering merely contact points and fingertip positions do not address this issue, while representations in the form of separate finger trajectories neglect correlating finger movements.

Hence, this work proposes a task space representation which incorporates synergies between the fingers by means

M. Do, T. Asfour and R. Dillmann are with the Karlsruhe Institute of Technology (KIT), Karlsruhe, Germany, as members of the Institute for Anthropomatics, e-mail: martin.do,asfour,dillmann@kit.edu

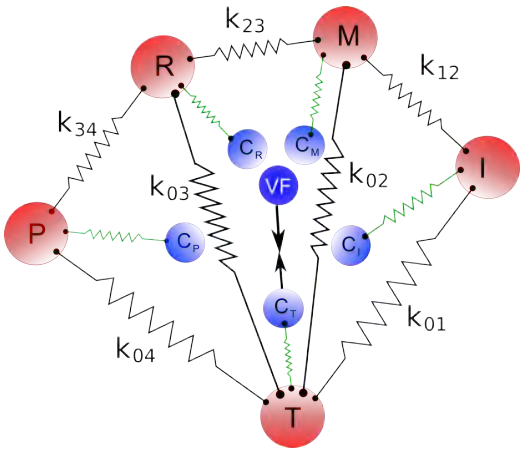


Fig. 1. Visualization of the bodies representing the fingertips. The virtual springs are visualized in black and denoted by the corresponding spring constant k_{ij}. VF describes the virtual finger while thumb (T), index (I), middle (M), ring (R) and pinkie (P) are indexed with 0, 1, 2, 3, 4.

of mass-spring-damper systems whose parameters form the representation of a grasp. Furthermore, we will show that this representation can be an initial building block for a grasp imitation learning framework which allows the parameter estimation from human observation, mapping to a humanoid platform, and the execution by reaching and grasping.

The paper is organized as follows. Section II describes the proposed representation of a grasp consisting of the model of the fingertip motion and the reach movement. In Section III, the experimental setup is explained containing a description of the humanoid platform, the observation mechanism and mapping to the robot platform. Finally, experimental results are given in IV. In conclusions, the work is summarized and notes to future works are given.

II. Representation of a Grasp

A. Concept

In this work, we suggest a representation which exploits synergies in task space by exploring and modulating fingertip movements during the grasp process. To establish synergies, one has to ensure that the trajectory of each fingertip is influenced by the motion of the remaining fingers, especially the neighboring ones. To model these relationships, ordinary mass-spring-damper systems are introduced as virtual springs between the fingertips as depicted in Fig. 1. The motion at time t of finger i at position $\mathbf{p_i} \in \mathbb{R}^{dim}$ connected to a finger j at position $\mathbf{p_j} \in \mathbb{R}^{dim}$ via a virtual spring can be inferred from following second order system of differential equations:

$$\ddot{\mathbf{x}}_{\mathbf{ij}}(t) = -\frac{k_{ij}}{m_i}\mathbf{x}_{\mathbf{ij}}(t) - \frac{d}{m_i}\dot{\mathbf{x}}_{\mathbf{ij}}(t), \tag{1}$$

with

$$\mathbf{x}_{\mathbf{ij}}(t) = \frac{\mathbf{p_i} - \mathbf{p_j}}{\|\mathbf{p_i} - \mathbf{p_j}\|}(\|\mathbf{p_i} - \mathbf{p_j}\| - l_{ij}), \tag{2}$$

describing the displacement concerning the equilibrium length l_{ij} along the spring direction and $\dot{\mathbf{x}}_{\mathbf{ij}}(t)$ and $\ddot{\mathbf{x}}_{\mathbf{ij}}(t)$ representing the corresponding velocity and acceleration. m_i denotes the mass of the physical body which represents the fingertip i while k_{ij} denotes the spring constant and d the damping constant. Auxiliary springs, which link each finger to its supposed contact position $\mathbf{c_i}$ are added to the system reducing oscillations in order to maintain stability and to retain the system centered around the contact points. The forces of each auxiliary spring can be determined by following equation:

$$\ddot{\mathbf{x}}_{\mathbf{c_i}}(t) = -k_c\mathbf{x}_{\mathbf{c_i}}(t) - d\dot{\mathbf{x}}_{\mathbf{c_i}}(t), \tag{3}$$

where $\mathbf{x}_{\mathbf{c_i}}(t)$ is obtained by replacing $\mathbf{p_j}$ with $\mathbf{c_i}$ in Eq. 2. k_c is constant and holds the same value for all auxiliary springs. In order to grasp an object, we desire a smooth, simultaneous movement of all finger towards their corresponding contact points. According to neuroscientific studies (see [14]), during the grasp process humans tend to focus on a mainly fixed spot on the object surface which corresponds to the thumb contact position. Therefore, the thumb is assumed to lead the reaching movement of the end effector towards the object. Following a concept introduced in [15], the remaining fingers form the virtual finger whose forces are supposed to build up an opposition force to the force exerted by the thumb in order to achieve a stable grasp. Based on these findings, to attain balanced, simultaneous finger movements, a central force $\mathbf{f_{cen}}(t)$ is applied on the entire system, which exerts a force $\mathbf{f_{i,ext}}(t)$ on each auxiliary spring resulting in:

$$\mathbf{f_{i,ext}}(t) = \begin{cases} \mathbf{f_{cen}}(t) & ,i = 1 \\ \left(\frac{\sum_{i=2}^{N}\|\mathbf{x_{c_i}(t)}\|}{(N-1)\|\mathbf{x_{c_1}(t)}\|}\right)^2 \mathbf{f_{cen}}(t) & ,else, \end{cases} \tag{4}$$

with finger $i = 1$ indexing the thumb. For the description of the entire system, we introduce a connection matrix $\mathbf{K}$ with $\mathbf{K}(i,j) = k_{ij}$ and a damping matrix $\mathbf{D}$ with $\mathbf{D}(i,j) = d_{ij}$ where $k_{ij} > 0$, $d_{ij} = d$ if two fingers are connected by virtual spring, and $k_{ij} = 0$, $d_{ij} = 0$ otherwise. Furthermore, a vector $\mathbf{c}$ is introduced indicating whether a finger is involved in a grasp by setting $c(i) = 1$, and $c(i) = 0$ otherwise. To obtain the complete equation for the motion of the body i, Eq. 3 and Eq. 4 are added to Eq. 1 which leads to following equation:

$$\ddot{\mathbf{x}}_{\mathbf{i}}(t) = -\mathbf{K}\mathbf{x}_{\mathbf{ij}}(t) - \mathbf{D}\dot{\mathbf{x}}_{\mathbf{ij}}(t) + \mathbf{c}^{\mathbf{T}}(\ddot{\mathbf{x}}_{\mathbf{c_i}}(t) + \mathbf{f_{i,ext}}(t)). \tag{5}$$

By solving Eq. 5 one obtains the displacements $\mathbf{x_i}(t)$ by which the position $\mathbf{p_i}$ is updated. Due to stability reasons the implicit fourth-order Runge-Kutta method is used as a solver.

In case contact points are not available, the proposed system can be applied on simple shaped objects by modifying the $\mathbf{x_{c_i}}(t)$. Replacing each contact point with the object center $\mathbf{c_o}$ and introducing the mean distance $\bar{d}(\{\mathbf{p_s}\}, \mathbf{c_o})$ between relevant surface points $\{\mathbf{p_s}\}$ and $\mathbf{c_o}$ as equilibrium length one obtains:

$$\bar{\mathbf{x}}_{\mathbf{c_i}}(t) = \frac{\mathbf{p_i} - \mathbf{c_o}}{\|\mathbf{p_i} - \mathbf{c_o}\|}(\|\mathbf{p_i} - \mathbf{c_o}\| - \bar{d}(\{\mathbf{p_s}\}, \mathbf{c_o})). \tag{6}$$

Applying Eq. 6 to Eq. 5 instead of Eq. 3 leads to the desired system equations. Relevant surface points can be extracted e. g. from a silhouette which one obtains when intersecting

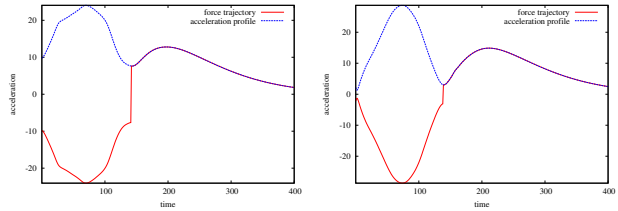


Fig. 2. Blue line, starting from positive, describes the acceleration for a reaching movement towards the object. The red line is the derived force trajectory. Left: Grasp from above the object. Right: Grasp from the side of the object.

a plane orthogonal to the reach direction of the end effector with a volumetric object representation.

Most of the parameters within the system are assumed to be constant or can be calculated except for the central force $\mathbf{f_{cen}}(t)$ whereas $\mathbf{f_{cen}}(t)$ plays a crucial role in the modulation of the system. For $\mathbf{f_{cen}}(t) < 0$ finger movements are generated which lead the fingertips away from the object whereas this process can be considered as the preshaping of the hand. The enclosing movement is initiated when $\mathbf{f_{cen}}(t) > 0$, causing the system to progress towards the final grasp pose. Therefore, $\mathbf{f_{cen}} = \{\mathbf{f_{cen}}(0) \dots \mathbf{f_{cen}}(t)\}$ can be interpreted as a force trajectory which controls the execution and transition of the different grasp phases. The trajectory can be inferred from the acceleration profile of the reach movement as follows:

$$\mathbf{f_{cen}}(t) = \begin{cases} -\|\mathbf{a}(t)\| & ,t < t_{a_{min}} \\ \|\mathbf{a}(t)\| & ,else, \end{cases} \tag{7}$$

where $t_{a_{min}}$ denotes the moment where the acceleration magnitude becomes minimal. The force trajectories for two different reach movements are depicted in Fig. 2.

B. Parameterization

The shape of the the finger trajectories emerging from the modulation of the system mainly depends on the spring constants of the virtual springs. For M springs the constants are collected in a vector $\mathbf{k} = (k_1, \dots, k_M)^T$. A major advantage of the proposed representation is that $\mathbf{k}$ can be estimated from the observation of the fingertip motion. For N fingers given their observed trajectory $\{\mathbf{p_i} = \{\mathbf{p_i}(0) \dots \mathbf{p_i}(t)\} | i = 1 \dots N\}$ and the force trajectory $\mathbf{f_{cen}}$ of the human hand, in order to estimate the springs constants, one has to rewrite Eq. 5 resulting in:

$$\mathbf{X_t k} = -\ddot{\mathbf{x}}_\mathbf{i}(t) - \mathbf{D}\dot{\mathbf{x}}_\mathbf{ij}(t) + \mathbf{c}^\mathbf{T}(\ddot{\mathbf{x}}_{\mathbf{c_i}}(t) + \mathbf{f_{i,ext}}), \tag{8}$$

where matrix $\mathbf{X_t} \in R^{(N \cdot dim) \times M}$ describes the displacements of each fingertip along the spring m:

$$\begin{aligned} X_t(i*dim+1, m) &= x_{m_i}(t) \\ &\vdots \\ X_t(i*dim+dim, m) &= x_{m_i}(t), \end{aligned} \tag{9}$$

with $n = 1, \dots, dim \cdot N$ and $x_{m_i}(t) = x_{ij}(t)$ if body i is connected to j via spring m, and $x_{m_i}(t) = 0$ otherwise. $\ddot{\mathbf{x}}_\mathbf{i}$ represents the accelerations of the fingers in task space calculated from the observed finger trajectories. Since Eq. 8 forms a linear regression problem, we apply Singular Value Decomposition to produce $\hat{\mathbf{X}}_\mathbf{t}^{-1}$, the generalized inverse matrix of $\mathbf{X_t}$. Subsequently, by multiplying $\hat{\mathbf{X}}_\mathbf{t}^{-1}$ on both sides an estimation for $\mathbf{k}$ is obtained.

C. Representation of the reach movement

For a complete grasp representation, the reach movement extracted from human observation has to be represented in a way, which allows the adaptation of learned action to new situations. We investigated different approaches for the representations of movement primitives based on splines [16], Hidden Markov Models [17] and applied dynamic motor primitives (DMP) as proposed in [18]. A DMP provides a representation of a movement segment by shaping an attractor landscape described by a second order dynamical system. In [19], a motion representation based on DMPs is applied to represent pick-and-place actions. Similar to a linear spring system, using second order dynamics the basic point attractive system can be written as follows:

$$\tau \dot{\mathbf{v}} = k(\mathbf{g} - \mathbf{x}) - d\mathbf{v} - k(\mathbf{g} - \mathbf{x_0})s + kf(s) \tag{10}$$

$$\tau \dot{\mathbf{x}} = \mathbf{v}, \tag{11}$$

where $\mathbf{x}$ and $\mathbf{v}$ are position and velocity of the system; $\mathbf{x_0}$ and $\mathbf{g}$ are the start and goal position; τ is a temporal scaling factor; k acts like a spring constant; the damping term d is chosen such that the system is critically damped. To enable the encoding of arbitrarily complex movements, the non-linear function f is introduced which is defined as follows:

$$f(s) = \frac{\sum_i w_i \psi_i(s) s}{\sum_i \psi_i(s)}, \tag{12}$$

where $\psi_i(s) = \exp(-h_i(s - c_i)^2)$ are Gaussian basis functions, with center c_i and width h_i, and w_i are adjustable weights. The function f depends on a phase variable s, which monotonically changes from 1 towards 0 during a movement and is obtained by following equation:

$$\tau \dot{s} = -\alpha s \ , \tag{13}$$

where α is a pre-defined constant. Eq. 13 is referred to as canonical system. Based on a demonstrated movement $\mathbf{x}(t)$ with time steps $t = 0, \dots, T$ and its corresponding velocity and acceleration profile $\mathbf{v}(t)$ and $\dot{\mathbf{v}}(t)$, a DMP can be adapted to a movement by adjusting the weights w_i within f. Since f can be computed from Eq. 11 with the demonstration parameters and s can be obtained through integrating the canonical system, the adjustment of w_i is reduced to a linear regression problem. The DMP formulation features several advantageous properties such as guaranteed convergence towards the goal, spatial and temporal invariance and robustness against perturbations. However, the most important property lies in the simple adaption towards a new situation, which is mainly accomplished by specifying new start and goal positions. Once specified, the execution of the movement is attained through integration and evaluation of $s(t)$. The obtained

Fig. 3. Left: The humanoid robot ARMAR-IIIb. Right: Position-controlled right hand with 8 DoF.

phase variable then drives the non-linear function f which in turn perturbs the linear spring-damper system to compute the desired attractor landscape. Regarding the concept introduced in Section II-A, the acceleration profile $\dot{\mathbf{v}}(t)$ of the DMP can be used to derive the force trajectory $\mathbf{f}_{\mathbf{cen}}$ as defined in Eq. 7. For each specific grasp type a DMP is generated and stored in motion library along with the parameters of the grasp representation.

III. Experiments

A. Experimental Platform

The humanoid robot ARMAR-IIIb, which serves as the experimental platform in this work, is a copy of the humanoid robot ARMAR-IIIa [20]. From the kinematics point of view, the robot consists of seven subsystems: head, left arm, right arm, left hand, right hand, torso, and a mobile platform. The head has seven DoF and is equipped with two eyes, which have a common tilt and can pan independently. Each eye is equipped with two digital color cameras, one with a wide-angle lens for peripheral vision and one with a narrow-angle lens for foveal vision. The upper body of the robot provides 33 DoF: 2·7 DoF for the arms and three DoF for the torso. The arms are designed in an anthropomorphic way: three DoF for each shoulder, two DoF in each elbow and two DoF in each wrist. Each arm is equipped with a pneumatic-actuated five-fingered hand with eight DoF. The locomotion of the robot is realized using a wheel-based holonomic platform.

B. Observation

In the following, a method for capturing the human fingertip motion is presented. Based on [21] and [22], we implemented a real-time tracking algorithm combining particle filter and mean shift based on color information. The input to the system is a stereo color image sequence, captured with the built-in wide-angle stereo pair of the humanoid robot ARMAR-IIIb. To obtain accurate and robust position estimations of the fingertips, markers in the form of green caps are attached to the fingers. In the first frame, the color information of the markers is exploited to segment the images in order to determine the regions of interest surrounding the N fingertips. These regions are labeled and a color histogram model in HSV space is calculated. A single particle filter instance is applied to obtain an estimation for all fingertip positions based on the previous observation and the weighted particles. Each particle represents a set of N candidate regions whereas the corresponding weight is calculated by comparing the regions color histograms to the histogram model and the posture of these candidates to the one in the previous frame. The estimation is refined using an ordinary mean shift algorithm driving each region towards the maxima of the density distribution within the color histogram. Since the markers are of the same color, overlaps and false labeling might occur. For grasp observation, the assumption is made that the palm is facing towards the camera where in most cases the finger order Thumb → Index → Middle → Ring → Pinkie is valid. By representing the coordinates in polar space, it can be checked easily if this order is violated. If so, a search for candidate regions for the false estimated fingers is initiated in the vicinity of the previous configuration. Since this algorithm operates on monocular images, for each view a tracking instance is created whereas the 3D positions of the fingers are calculated by exploiting epipolar geometry. The presented framework is capable of online tracking of fingertip motion with a frame rate of 23 Hz on a 2 GHz dual core CPU. Sample images during the tracking process are depicted in Fig. 4.

C. Mapping and Execution

The grasp reproduction on ARMAR-IIIb is performed in several stages. In the first stage, the DMP for the reaching movement is adapted to position and the orientation of the object to be grasped. Subsequently, the force trajectory is derived to modulate the grasp representation resulting in the fingertip trajectories. The trajectories are mapped and scaled to fit the coordinate system of an intermediate hand model. For this purpose, the Master Motor Map (MMM), introduced in [23] and extended in [24], is used. The core feature of this framework is a reference kinematic model which facilitates the mapping from a human motion capture system to the kinematic structure of a robot. The model incorporates a biomechanical hand model with 21 DoF. By solving the inverse kinematics problem for the MMM hand model, one obtains the joint angle configuration respective to the given fingertip positions. Due to different measurements and less DoF of the robot hand, in order to attain a goal-directed reproduction, which additionally features a high similarity to the demonstrated human hand movement, joint

Fig. 4. Left camera views of the tracking method. Red denotes thumb region, light blue the index, blue the middle, pink the ring finger, and red pinkie region

angles as well as the desired fingertip positions have to be considered during execution. In [25], we developed an approach, which supports reproduction of observed human motion on the robot using non-linear optimization methods. To formulate an optimization problem for each finger which comprises displacements in Cartesian space regarding the fingertip position as well as the finger joints, a similarity measure is defined as follows:

$$S(\sigma) = 2 - \frac{\frac{1}{n}\sum_{i=1}^{n}\left(\hat{\sigma}_i^t - \sigma_i\right)^2}{\pi^2} - \frac{\frac{1}{3}\sum_{k=1}^{3}\left(\hat{p}_k^t - p_k\right)^2}{\left(2 \cdot l_{finger}\right)^2} \qquad (14)$$

with n representing the number of finger joints, $\sigma_i, \hat{\sigma}_i^t \in [0, \pi]$ and $p_k, \hat{p}_k^t \in \left[-l_{finger}, l_{finger}\right]$, whereas l_{finger} describes the considered finger length. The reference joint angle configuration is denoted by $\hat{\sigma} \in \mathbb{R}^n$, while $\hat{\mathbf{p}} \in \mathbb{R}^3$ stands for the desired fingertip position. The current fingertip position $\mathbf{p}$ can be determined by applying the forward kinematics of the robot to the joint angle configuration σ. Based on Eq. 14 and the joint constraints $\{(C_{min}, C_{max})\}$ of a robot with n joints, one obtains following constrained optimization problem:

$$\min S'(\sigma) = 2 - S(\sigma) \qquad (15)$$

$$\text{subject to} \quad C_{i_{min}} \leq \hat{\sigma}_i \leq C_{i_{max}} \qquad (16)$$

For solving Eq. 15, we apply the Levenberg-Marquardt algorithm. Following this optimization approach a trade-off is attained, which on the one hand results in an accurate finger positioning with small displacement error while it provides on the other hand a feasible robot joint angle configuration resembling the observed human configuration.

D. Results

The N-body system of the grasp representation is implemented in 2D. Therefore, currently only planar grasps which only require fingertip contact can be represented. For the reproduction of grasp, the force trajectory defined in Eq. 7 is applied to modulate the systems. The trajectories emerging from this modulation describe a fingertip posture in x,y direction in task space of the hand. The pose of the hand needed for grasping is obtained from the DMP movement and represented in the robot's platform coordinates. The proposed grasp representation is evaluated for a pinch, tripod, power, and lateral grasp. Based on image sequences captured by the humanoid robot, in addition to the fingertip trajectories the hand movement was determined by segmenting and tracking the hand by means of skin color information. From the resulting trajectory, a DMP is generated which, due its properties, allows the adaptation to new targets and the reproduction of smooth trajectories. The results of the motion reproduction of are depicted in Fig. 5.

Based on the fingertip trajectories, it is possible to estimate the spring constants of each virtual spring. The constants of the remaining springs are fixed independently of the considered grasp type. To maintain stability and avoid oscillation during the modulation, the system is assumed to be over-damped. For our experiments, the trajectories emerging

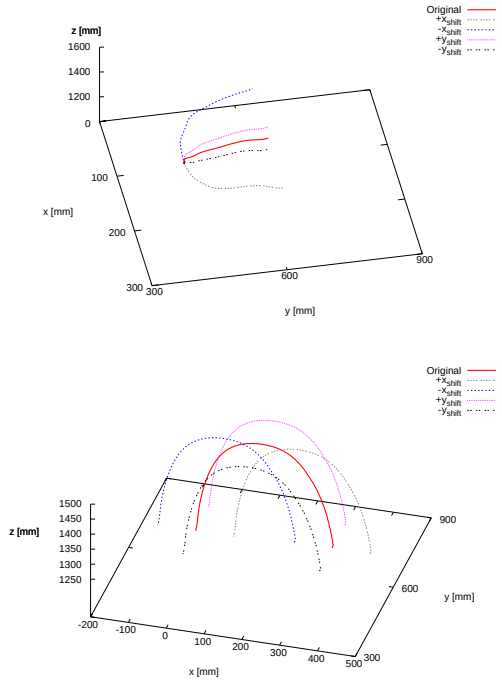


Fig. 5. Top: From a DMP reproduced reach trajectories towards different goals. Bottom: From a DMP reproduced place trajectories starting from different start points towards different goals.

from a grasp instantiations are compared to the observed movements. Due to the small number of contact points, the fingertip movements during a pinch grasp reproduction are highly informative in terms similarity to the human demonstration. As depicted in Fig. 6, using the virtual spring grasp representation, for thumb, index, middle, and ring finger, fingertip movements could be generated which are similarly shaped as the observed trajectories. Due to noisy motion data regarding the pinkie movement, the spring constants linked to the pinkie could not be estimated accurately enough leading to a slightly diverse trajectory. Furthermore, it could be observed that due to the integration of springs, we were able to produce smooth finger trajectories. To complete the specification of the grasp representation, the equilibrium lengths between the fingers were measured at a human subject whereas the hand is to be maintained in a very relaxing posture. On the platform, we were able to reproduce grasping movements where the hand preshapes and contact with the object is made at the end of the reaching movement. However, due to small number of DoF of the ARMAR-IIIb hand and its unstable pneumatic control mechanism, the optimized joint angle configuration could not be accurately reproduced. Results of the grasp reproduction for a pinch and power grasp are depicted in Fig. 8 and Fig. 7.

IV. Conclusion

In this work, we have presented a grasp representation which exploits finger movement synergies in task space and, hence, allows the formulation of grasps in a goal-directed and

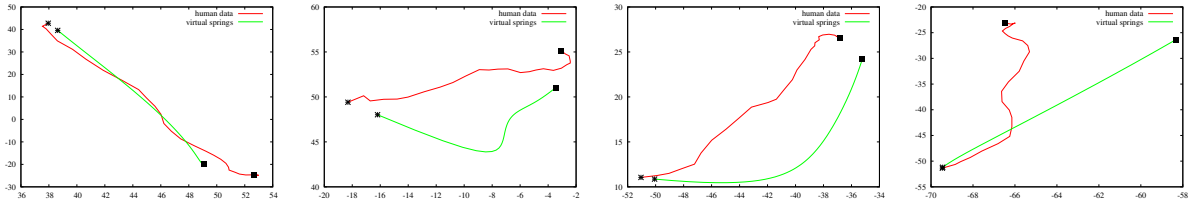


Fig. 6. Plots showing the trajectory of each finger during observation (red) and reproduction (green) of a pinch grasp. The star symbol denotes the start of each trajectory, whereas the box symbol represents the end. The measurements are given in mm. From left to right: index finger; middle finger; ring finger; pinkie.

low dimensional fashion facilitating several processes such as the observation of the human hand, which is a cumbersome task in joint space. Along with the parameter estimation procedure, a grasp can be learned and represented from human demonstration, even online. In order to reduce the dimensions of the control variables for trajectory generation, synergies on task space level were successfully established by means of the virtual springs. The result is a continuous grasp representation, which unifies the different grasp stages (preshape, reach, and enclose) leading to a smooth, human-like movement reproduction. In the near future, we focus on extending our implementation of the dynamical system from 2D to 3D in order to represent grasps which require additional contact areas besides the fingertips. Furthermore, we will extend our library of represented grasps and investigate how complex object representations can be integrated.

V. Acknowledgments

The work described in this paper was conducted within the EU Cognitive Systems project GRASP (FP7- 215821) funded by the European Commission.

References

[1] J. Napier, "The prehensile movements of the human hand," *Journal of Bone and Joint Surgery*, vol. 38B, no. 4, p. 902913, 19.

[2] M. R. Cutkosky and P. K. Wright, "Modeling manufacturing grips and correlation with the design of robotic hands."

[3] A. T. Miller, S. K. H. T. Christensen, and P. K. Allen, "Automatic grasp planning using shape primitives," in *Proc. IEEE International Conference on Robotics and Automation (ICRA'03)*, Taipei, Taiwan, September 2003, pp. 1824–1829.

[4] M. Do, J. Romero, H. Kjellström, P. Azad, T. Asfour, D. Kragic, and R. Dillmann, "Grasp recognition and mapping on humanoid robots," in *IEEE International Conference on Humanoid Robots*, Paris, France, December 2009.

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

[6] M. T. Ciocarlie and P. K. Allen, "Hand posture subspaces for dexterous robotic grasping," *The International Journal of Robotics Research*, vol. 28, no. 7, pp. 851–867, 2009.

[7] M. Gabiccini and A. Bicchi, "On the role of hand synergies in the optimal choice of grasping forces," in *Proceedings of Robotics Science and Systems*, Zaragoza, Spain, June 2010.

[8] D. Pratichizzo, M. Malvezzi, and A. Bicchi, "On motion and force controllability of grasping hands with postural synergies," in *Proceedings of Robotics Science and Systems*, Zaragoza, Spain, June 2010.

[9] J. Steffen, R. Haschke, and H. Ritter, "Towards dextrous manipulation using manipulation manifolds," in *Proc. IEEE International Conference on Intelligent Robots and Systems (IROS'08)*, Nice, France, September 2008, pp. 2738–2743.

[10] B. Hoff and M. A. Arbib, "Models of Trajectory Formation and Temporal Interaction of Reach and Grasp," *Journal of Motor Behaviour*, vol. 25, pp. 175–192, 1993.

[11] M. Jeannerod, M. A. Arbib, G. Rizzolatti, and H. Sakata, "Grasping objects: the cortical mechanisms of visuomotor transformation," *Trends in Neurosciences*, vol. 18, no. 7, pp. 314–320, 1995.

[12] N. Vahrenkamp, M. Do, T. Asfour, and R. Dillmann, "Integrated grasp and motion planning," in *Proc. IEEE International Conference on Robotics and Automation (ICRA'10)*, Taipei, Taiwan, 2010, pp. 2883–2888.

[13] S. Ekvall and D. Kragic, "Interactive grasp learning based on human demonstration," in *Proc. IEEE International Conference on Robotics and Automation (ICRA'04)*, New Orleans, USA, April 2004, pp. 3519–3524.

[14] A. Schiegg, H. Deubel, and W. X. Schneider, "Attentional selection during preparation of prehension movements," *Visual Cognition*, vol. 10, pp. 409–431, 2003.

[15] M. Arbib, T. Iberall, and D. Lyons, "Coordinated control programs for movements of the hand," *Experimental Brain Research Supplement*, vol. 10, pp. 111–129, 1985.

[16] A. Ude, C. G. Atkeson, and M. Riley, "Programming Full-Body Movements for Humanoid Robots by Observation," *Robotics and Autonomous Systems*, vol. 47, no. 2-3, pp. 93–108, June 2004.

[17] T. Asfour, P. Azad, F. Gyarfas, and R. Dillmann, "Imitation Learning of Dual-Arm Manipulation Tasks in Humanoid Robots," *International Journal of Humanoid Robotics*, vol. 5, no. 2, pp. 183–202, December 2008.

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

[19] P. Pastor, H. Hoffmann, T. Asfour, and S. Schaal, "Learning and generalization of motor skills by learning from demonstration," in *Proceedings of the IEEE International Conference on Robotics and Automation*, Kobe, Japan, 2009.

[20] T. Asfour, K. Regenstein, P. Azad, J. Schröder, N. Vahrenkamp, and R. Dillmann, "ARMAR-III: An Integrated Humanoid Platform for Sensory-Motor Control," in *IEEE/RAS International Conference on Humanoid Robots*, 2006.

[21] E. Maggio and A. Cavallaro, "Hybrid particle filter and mean shift tracker with adaptive transition model," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2005.

[22] S. Yonemoto and M. Sato, "Multitarget Tracking using Mean-Shift with Particle filter based intialization," in *12th International Conference on Information Visualisation*, 2008, pp. 314–320.

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

[24] S. Gärtner, M. D. Azad, T. Asfour, R. Dillmann, C. Simonidis, and W. Seemann, "Generation of Human-like Motion for Humanoid Robots Based on Marker-based Motion Capture Data," in *ISR 2010 (41st Internationel Symposium on Robotics*, Munich, Germany, June 2010.

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

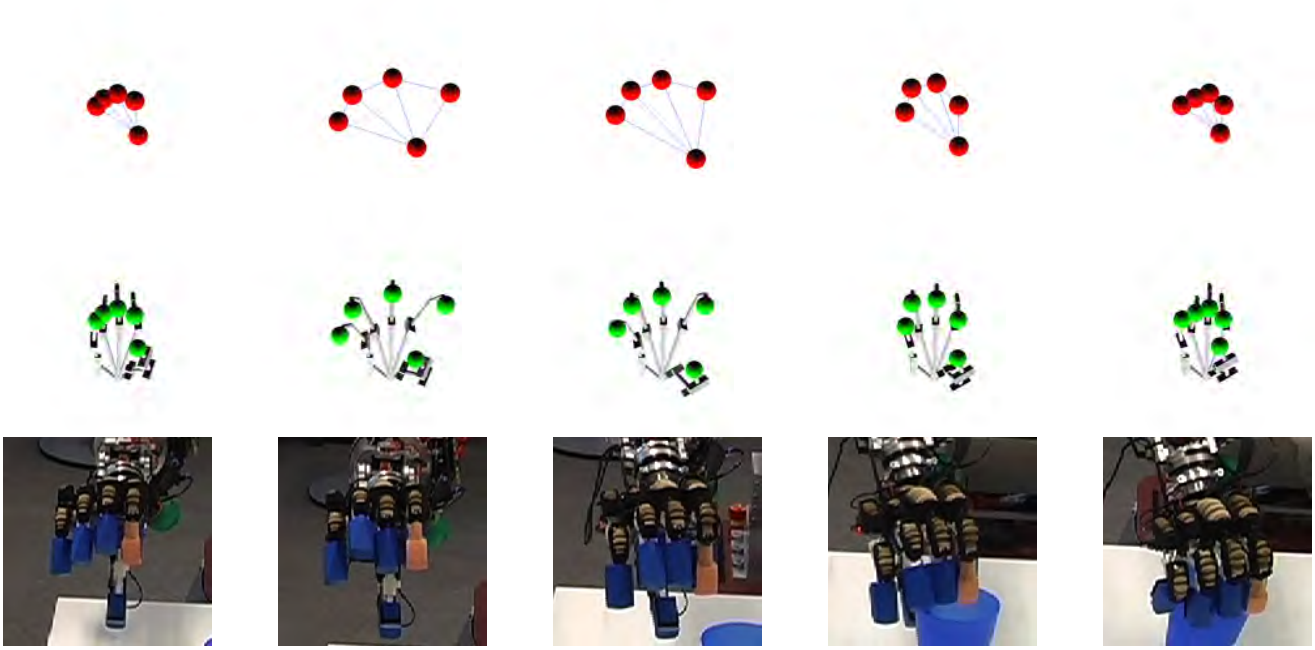


Fig. 7. Top: Image sequence depicting the capture human fingertip motion for a power grasp. Middle: The MMM hand model maintaining a hand posture that matches the fingertip motion. Bottom: Reproduction of the represented power grasp.

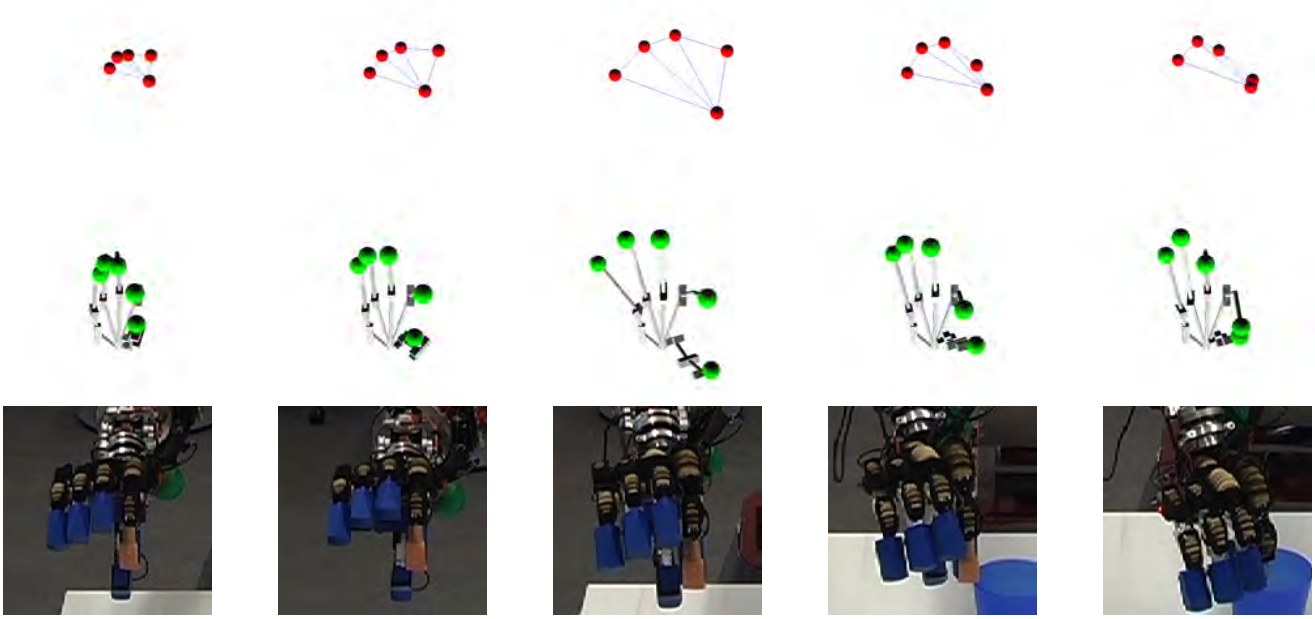


Fig. 8. Top: Image sequence depicting the capture human fingertip motion for a pinch grasp. Middle: The MMM hand model maintaining a hand posture that matches the fingertip motion. Bottom: Reproduction of the represented pinch grasp.

Visual Servoing on Unknown Objects

Xavi Gratal, Javier Romero, Jeannette Bohg, Danica Kragic

Computer Vision and Active Perception Lab
Centre for Autonomous Systems
School of Computer Science and Communication
Royal Institute of Technology, 10044 Stockholm, Sweden
{javiergm,jrgn,bohg,danik}@nada.kth.se

Abstract

We study visual servoing in a framework of detection and grasping of unknown objects. Classically, visual servoing has been used for applications where the object to be servoed on is known to the robot prior to the task execution. In addition, most of the methods concentrate on aligning the robot hand with the object without grasping it. In our work, visual servoing techniques are used as building blocks in a system capable of detecting and grasping unknown objects in natural scenes. We show how different visual servoing techniques facilitate a complete grasping cycle.

Keywords: visual servoing, object grasping, calibration, active vision

1. Introduction

Object grasping and manipulation stands as an open problem in the area of robotics. Many approaches assume that the object to be manipulated is known before hand [1, 2, 3, 4]. If the object bears some resemblance to a known object, experience can be used for grasp synthesis [5, 6, 7, 8]. An unknown object needs to be analyzed in terms of its 3D structure and other physical properties from which a suitable grasp can be inferred [9, 10, 11, 12].

Realistic applications require going beyond open-loop execution of these grasps and the ability to deal with different type of errors occurring in an integrated robotic system. One kind of errors are systematic and repeatable, introduced mainly by inaccurate kinematic models. These can be minimized offline through precise calibration. The second kind are random errors introduced by a limited repeatability of the motors or sensor noise. These have

to be compensated through online mechanisms.

In this paper, we show how *Visual Servoing* (VS) can be used at different stages in a grasping pipeline to correct errors both offline and online. One of the contributions is the application of VS for automatic calibration of the hardware. We follow the classical approach of applying VS for aligning the robot hand with the object prior to grasping it [13]. This requires tracking of the manipulator pose relative to the camera. Instead of the common approach of putting markers on the robot hand, we use a model based tracking system. This is achieved through *Virtual Visual Servoing* (VVS) in which the systematic and random errors are compensated for. An additional contribution is the generalisation of VVS to CAD models instead of using object models tailor-made for the application.

The remainder of this paper is organised as follows. Section 2 formalises the systematic and random errors inherent to the different parts of the robotic system. In Section 3, related work in the area of offline calibration as well as closed-loop control is presented. The approach proposed in this paper is described in detail in Section 4. Its performance is analysed quantitatively on synthetic data and qualitatively on our robotic platform. The results of these experiments are presented in Section 5.

2. Problem Formulation

Object grasping in real world scenarios requires a set of steps to be performed prior to the actual manipulation of an object. A general outline of our grasping pipeline is provided in Figure 2. The hardware components of the system as shown in Figure 1 are (i) the Armar III robotic head [14] equipped with two stereo camera pairs (wide-angle for peripheral vision and narrow-angle for foveal vision), (ii) the 6 DoF Kuka arm KR5 sixx R850 [15] and (iii) the Schunk Dexterous Hand 2.0 (SDH) [16].

2.1. Grasping Pipeline

The main pre-requisite for a robot to perform a pick-and-place task is to have an understanding of the 3D environment it is acting in. In our pipeline, we perform scene construction by using the active head for visual exploration and stereo reconstruction as described in detail in our previous work [17]. The resulting point cloud is segmented into object hypotheses and background.

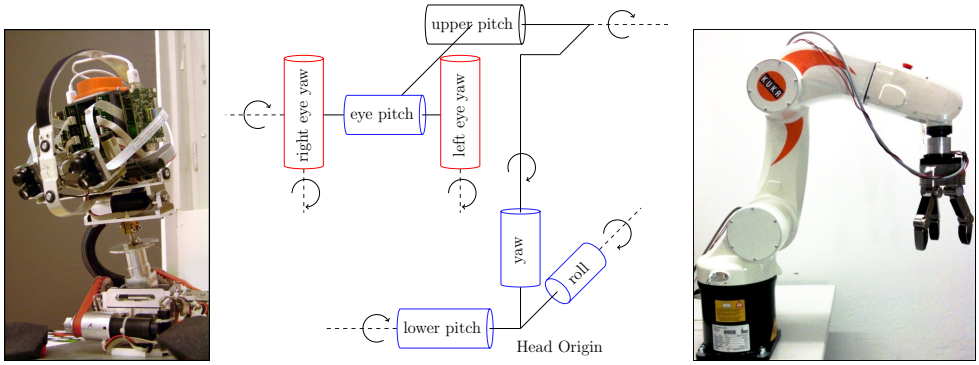


Figure 1: Hardware Components in the Grasping Pipeline. 1(a) Armar III Active Head with 7 DoF. 1(b) The Kinematic Chain of the Active Head. The upper pitch is kept static. Right and left eye yaw are actuated during fixation and thereby change the vergence angle and the epipolar geometry. All the other joints are used for gaze shifts. 1(c) Kuka Arm with 6 DoF and Schunk Dexterous Hand 2.0 (SDH) with 7 DoF.

The scene model then consists of the arm and the active head positioned relative to each other based on the offline calibration as described in Section 4.4. Furthermore, a table plane is detected and the online detected object hypotheses are placed on it.

Grasp inference is then performed on each hypothesis. For given grasp candidates, a collision-free arm trajectory is planned in the scene model and applied to the object hypothesis with the real arm.

2.2. Error Formalisation

The aforementioned grasping pipeline relies on the assumption that all the parameters of the system are perfectly known. This includes e.g. internal and external camera parameters as well as the pose of the head, arm and hand in a globally consistent coordinate frame.

In reality however two different types of errors are inherent to the system. One contains the systematic errors that are repeatable and arise from inaccurate calibration or inaccurate kinematic models. The other group comprises random errors originating from noise in the motor encoders or in the camera. These errors propagate and deteriorate the relative alignment between hand and object. The most reliable component of the system is the Kuka arm that has a repeatability of less than 0.03 mm [15]. In the following, we will analyse the different error sources in more detail.

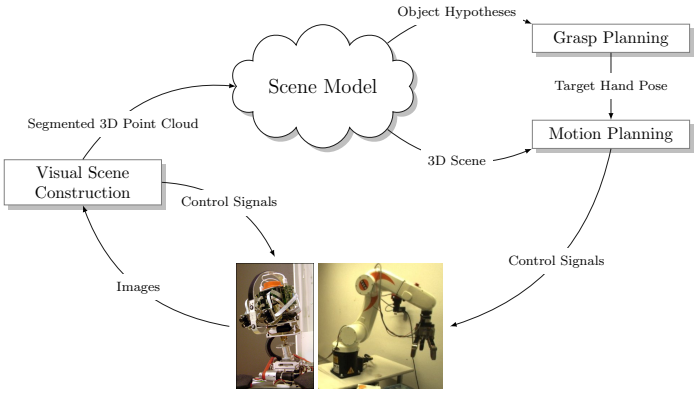


Figure 2: Our Open-Loop Grasping Pipeline.

2.2.1. Stereo Calibration Error

Given a set of 3D points $Q^W = \begin{bmatrix} x^W & y^W & z^W \end{bmatrix}^{\mathrm{T}}$ in the world reference frame W and their corresponding pixel coordinates $P = \begin{bmatrix} u & v & 1 \end{bmatrix}^{\mathrm{T}}$ in the image, we can determine the internal parameters $\mathbf{C}$ and external parameters $[\mathbf{R}_W^C | \mathbf{t}_W^C]$ for all cameras C in the vision system. This is done through standard methods by exploiting the following relationship between Q^W and P:

$$wP = \begin{bmatrix} wu \\ wv \\ w \end{bmatrix} = \mathbf{C}P^C \text{ with } P^C = \begin{bmatrix} z^C \\ y^C \\ z^C \end{bmatrix} = \mathbf{R}_W^C \left[P^W - \mathbf{t}_W^C \right] \tag{1}$$

Once we have these parameters for the left camera L and right camera R, the epipolar geometry as defined by the essential matrix $\mathbf{E} = \mathbf{R}_L^R [\mathbf{t}_L^R]_\times$ can be determined. Here $\mathbf{t}_L^R$ defines the baseline and $\mathbf{R}_L^R$ the rotation between the left and right camera system.

Our calibration method, which uses the arm as world reference frame, will be described in Section 4.4. The average reprojection error is 0.1 pixels. Since peripheral and foveal cameras are calibrated simultaneously, we also get the transformation between them.

Additionally to the systematic error in the camera parameters, other effects such as camera noise and specularities lead to random error in the stereo matching.

2.2.2. Positioning Error of the Active Head

We are using the robot head to actively explore the environment through gaze shifts and fixation on objects. This involves dynamically changing the epipolar geometry between the left and right camera system. Also the camera position relative to the head origin is changed. The internal parameters remain static.

The kinematic chain of the active head is shown in Figure 1(b). Only the last two joints, the left and right eye yaw, are used for fixation and thereby affect the stereo calibration. The remaining joints are actuated for performing gaze shifts.

In order to accurately detect objects with respect to the camera, the relation between the two camera systems as well as between the cameras and the head origin needs to be determined after each movement. Ideally, these transformations should be obtainable just from the known kinematic chain of the robot and the readings from the encoders. In reality, these readings are erroneous due to noise and inaccuracies in the kinematic model.

Let us first consider the epipolar geometry and assume that the left camera system defines the origin and remains static. Only the right camera rotates around its joint by ϕ at time k. According to the kinematic model this movement can be expressed by

$$[\mathbf{R}_{k-1}^{k}|\mathbf{t}_{k-1}^{k}] \text{ with } \mathbf{R}_{k-1}^{k} = \begin{bmatrix} \cos\phi & -\sin\phi & 0 \\ \sin\phi & \cos\phi & 0 \\ 0 & 0 & 1 \end{bmatrix} \text{ and } \mathbf{t}_{k-1}^{k} = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}^{\mathrm{T}} \quad (2)$$

which is a pure rotation around the z-axis of the joint. The new essential matrix would then be

$$\mathbf{E}_k = \mathbf{R}_k \mathbf{R}_L^R [\mathbf{t}_L^R]_\times. \quad (3)$$

Inaccuracies arise in the manufacturing process influencing the true center and axis of joint rotations and in the discrepancy between motor encoder readings and actual angular joint movement. This is illustrated in Figure 3 showing the translational component $\delta = \begin{bmatrix} \delta_x & \delta_y & \delta_z \end{bmatrix}^{\mathrm{T}}$ and rotational component $\epsilon = \begin{bmatrix} \epsilon_x & \epsilon_y & \epsilon_z \end{bmatrix}^{\mathrm{T}}$ of the error between the ideal and real joint positions. Under the assumption that only small angle errors occur, the error matrix can be approximated as [18]

$$[\mathbf{R}_e|\mathbf{t}_e] \text{ with } \mathbf{R}_e = \begin{bmatrix} 1 & -\epsilon_z & \epsilon_y \\ \epsilon_z & 1 & -\epsilon_x \\ -\epsilon_y & \epsilon_x & 1 \end{bmatrix} \text{ and } \mathbf{t}_e = \begin{bmatrix} \delta_x & \delta_y & \delta_z \end{bmatrix}^{\mathrm{T}} \quad (4)$$

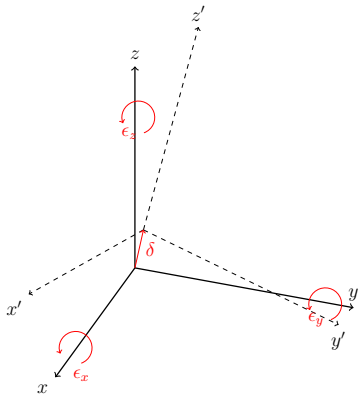


Figure 3: Six parametric errors in a rotary joint around the z-Axis. $\delta = [\delta_x\ \delta_y\ \delta_z]^{\mathrm{T}}$ is the translational component and $\epsilon = [\epsilon_x\ \epsilon_y\ \epsilon_z]^{\mathrm{T}}$ is the rotational component of the error.

Based on this, the true essential matrix can be modelled as

$$\mathbf{E}_k = \mathbf{R}_e \mathbf{R}_k \mathbf{R}_L^R [\mathbf{R}_e \mathbf{t}_L^R + \mathbf{t}_e]_\times . \tag{5}$$

Leaving this specific error matrix unmodelled will lead to erroneous depth measurements of the scene.

Similarly to the vergence angle, error matrices can be defined for every joint modelling inaccurate positioning and motion. Let us define J_{n-1} and J_n as two subsequent joints. According to the Denavit-Hartenberg convention, the ideal transformation $\mathbf{T}_{n-1}^n$ between these joints is defined as

$$\mathbf{T}_{n-1}^n = {}_z\mathbf{T}_{n-1}^n(d, \phi)\ \ {}_x\mathbf{T}_{n-1}^n(a, \alpha) \tag{6}$$

where ${}_z\mathbf{T}_{n-1}^n(d, \phi)$ describes the translation d and rotation ϕ with respect to the z-axis of J_{n-1}. ${}_x\mathbf{T}_{n-1}^n(a, \alpha)$ describes the translation a and rotation α with respect to the x-axis of J_n. While d, a and α are defined by the kinematic model of the head, ϕ is varying with the motion of the joints and can be read from the motor encoders.

The true transformation ${}_e\mathbf{T}_{n-1}^n$ will however look different. Modelling the position error $\mathbf{T}_{pe}$ and motion error $\mathbf{T}_{me}$ as in Equation 4 yields

$${}_e\mathbf{T}_{n-1}^n = {}_z\mathbf{T}_{n-1}^n(d, \phi)\mathbf{T}_{me}\ \ {}_x\mathbf{T}_{n-1}^n(a, \alpha)\mathbf{T}_{pe}. \tag{7}$$

These errors propagate through the kinematic chain and mainly affect the x and y position of points relative to the world coordinate system.

Regarding random error, the last five joints in the kinematic chain achieve a repeatability in the range of ± 0.025 degrees [14]. The neck pitch and neck roll joints in Figure 1(b) achieve a repeatability in the range of ± 0.13 and ± 0.075 degrees respectively.

2.2.3. Positioning Error of the Cameras with Respect to the Arm

As will be described in Section 4.4, we are using the arm to calibrate the stereo system. Therefore, assuming that the transformation from the head origin to the cameras is given, the error in the transformation between the arm and cameras is equivalent to the error of the stereo calibration.

3. Related Work

3.1. Closed-Loop Control in Robotic Grasping

In the previous section, we summarised the errors in a grasping system that lead to an erroneous alignment of the end effector with an object. A system that executes a grasp in closed loop without any sensory feedback is likely to fail.

In [19, 20] this problem is tackled by introducing haptic and force feedback into the system. Control laws are defined that adapt the pose of the end effector based on the readings from a force-torque sensor and contact location on the haptic sensors. A disadvantage of this approach is that the previously detected object pose might change during the alignment process.

Other grasping systems make use of visual feedback to correct the wrong alignment before contact between the manipulator and the object is established. Examples are proposed by Huebner et al. [2] and Ude et al. [21], who are using a similar robotic platform to ours including an active head. In [2], the Armar III humanoid robot is enabled to grasp and manipulate known objects in a kitchen environment. Similar to our system [17], a number of perceptual modules are at play to fulfill this task. Attention is used for scene search. Objects are recognized and their pose estimated with the approach originally proposed in [4]. Once the object identity is known, a suitable grasp configuration can be selected from an offline constructed database. Visual servoing based on a wrist spherical marker is applied to bring the robotic hand to the desired grasp position [22]. Different from our approach, absolute 3D data is estimated by fixing the 3 DoF for the eyes to a position for which a stereo calibration exists. The remaining degrees of freedom controlling the neck of the head are used to keep the target and current hand

position in view. In our approach, we reconstruct the 3D scene by keeping the eyes of the robot in constant fixation on the current object of interest. This ensures that the left and right visual field overlap as much as possible, thereby maximizing e.g. the amount of 3D data that can be reconstructed. However, the calibration process becomes much more complex.

In the work by Ude et al. [21], fixation plays an integral part of the vision system. Their goal is however somewhat different from ours. Given that an object has been already placed in the hand of the robot, it moves it in front of its eyes through closed loop vision based control. By doing this, it gathers several views from the currently unknown object for extracting a view-based representation that is suitable for recognizing it later on. Different to our work, no absolute 3D information is extracted for the purpose of object representation. Furthermore, the problem of aligning the robotic hand with the object is circumvented.

3.2. Calibration Methods

In [23], the authors presented a method for calibrating the active stereo head. The correct depth estimation of the system was demonstrated by letting it grasp an object held in front of its eyes. No dense stereo reconstruction has been shown in this work.

Similarly, in [24] a procedure for calibrating the Armar III robotic head was presented. Our calibration procedure is similar to the one described in those papers, with a few differences. We extend it to the calibration of all joints, thus obtaining the whole kinematic chain. Also, the basic calibration method is modified to use an active pattern instead of a fixed checkerboard, which has some advantages that we outline in Section 4.4.

3.3. Visual and Virtual Visual Servoing

For the accurate control of the robotic manipulator using visual servoing, it is necessary to know its position and orientation (*pose*) with respect to the camera. In the systems mentioned in Section 3.1, the end effectors of the robots are tracked based on fiducial markers like LEDs, colored spheres or Augmented Reality tags. A disadvantage of this marker based approach is that the mobility of the robot arm is constrained to keep the marker always in view. Furthermore, the position of the marker with respect to the end effector has to be known exactly. For these reasons, we propose to track the whole manipulator instead of only a marker. Thereby, we alleviate the problem of constrained arm movement. Additionally, collisions with the

object or other obstacles can be avoided in a more precise way. In this paper we track the pose of the robotic arm and hand assuming their kinematic chain to be perfectly calibrated, although the system can be adapted to estimate deviations in the kinematic chain.

Tracking objects of complex geometry is not a new problem and approaches can be divided into two groups: appearance-based and model-based methods. The first approach is based on comparing camera images with a huge database of stored images with annotated poses [25]. The second approach relies on the use of a geometrical model (3D CAD model) of the object and perform tracking based on optical flow [26]. There have also been examples that integrate both of these approaches [27].

Apart from the tracking itself, an important problem is the initialization of the tracking process. This can be done by first localizing the object in the image followed by a global pose estimation step. In our previous work, we have also demonstrated how the initialization can be done for objects in a generic way [28]. In the case of a manipulator, a rough estimate of its pose in the camera frame can be obtained from the kinematics of the arm and the hand-eye calibration. In [29], it has been shown that the error between this first estimate and the real pose of the manipulator can be corrected through virtual visual servoing, using a simple wireframe model of the object. In our work, a synthetic image of the robot arm is rendered based on a complete 3D CAD model and its initial pose estimate is compared and aligned with the real image.

In our recent work [30], we demonstrate how pose estimation can be performed for a complex articulated object such as a human hand. The major contribution of that work is the use of a discriminative machine learning approach for obtaining real-time tracking of an object with 27 degrees of freedom. The problem considered in this paper has a lower dimensionality and a more accurate guess of the initial pose. This makes it possible to adopt a real-time generative approach that renders the last pose of the object in each frame and estimates the new pose through a process of error minimization.

4. The Proposed System

For overcoming the problem of lacking a globally consistent coordinate frame for objects, manipulator and cameras, we introduce visual servoing into the grasping pipeline. This allows us to control the manipulator in closed

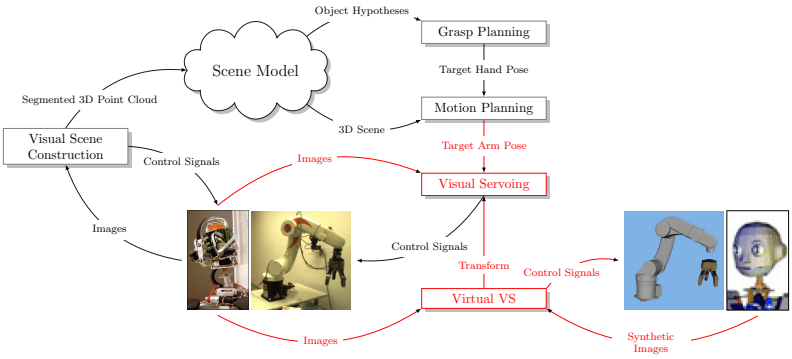


Figure 4: A Grasping Pipeline With Closed Loop Control of the Manipulator through Visual Servoing that is initialised through Virtual Visual Servoing. Armar III Head Model adapted from [31].

loop using visual feedback to correct any misalignment with the object online. We then use the camera coordinate frame as the global reference system in which the manipulated object and robot are also defined.

For accurately tracking the pose of the manipulator, we use virtual visual servoing. The initialisation of this method is based on the known joint values of hand and arm and the hand-eye calibration, which is obtained offline.

The adapted grasping pipeline is summarised in Figure 4. What follows is a more detailed description of all the components of this pipeline.

4.1. Vision System for Constructing the Scene Model

As described earlier, the scene model in which grasping is performed consists of a robot arm, hand and head as well as a table plane onto which object hypotheses are placed. The emergence of these hypotheses is triggered by the visual exploration of the scene with the robotic head. In the following, we will give a brief summary of this exploration process. More details can be found in our previous work [3, 32, 33, 17].

The active robot head has two stereo camera pairs. The wide-field cameras of which an example can be seen in Figure 5(a) are used for scene search. This is done by computing a saliency map on them and assuming that maxima in this map are initial object hypotheses (Figure 5(b)). A gaze shift is performed to a maxima such that the stereo camera with the narrow-angle lenses center on the potential object. An example of an object fixated in the foveal view is shown in Figure 5(c). In the following we will label the camera

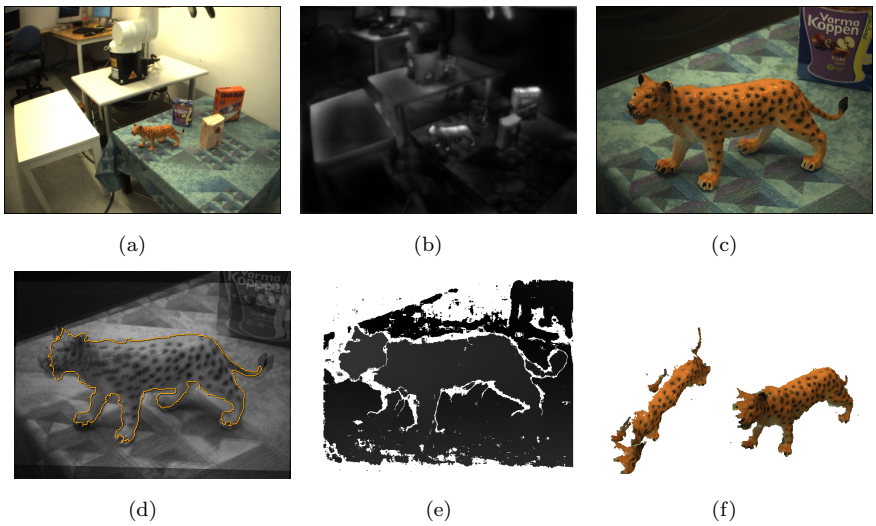


Figure 5: Example Output for Exploration Process. 5(a) Left Peripheral Camera. 5(b) Saliency Map of Peripheral Image. 5(c) Left Foveal Camera. 5(d) Segmentation on Overlayed Fixated Rectified Left and Right Images. 5(e) Disparity Map. 5(f) Point Cloud of Tiger.

pose when fixated on the current object of interest as C_0. Once the system is in fixation, a disparity map is calculated and segmentation performed (see Figure 5(e) and 5(d)). For each object, we then obtain a 3D point cloud (an example is shown in Figure 5(f)).

4.2. Grasp and Motion Planning

Once a scene model has been obtained by the vision system, a suitable grasp can be selected for each object hypothesis depending on the available knowledge about the object. In our previous work, we presented grasp planners on known, familiar or unknown objects [2, 5, 31, 17]. In [2, 31] resulting grasp hypotheses are tested in simulation on force closure prior to execution. The set of stable grasps is then returned to plan corresponding collision free arm trajectories.

The focus of this paper is the application of visual servoing and virtual visual servoing in the grasping pipeline. We have therefore selected a simple top-grasp selection mechanism that has been proven to be very effective for pick-and-place tasks in table-top scenarios [17]. For this, we calculate the eigenvectors and centre of mass of the projection of the point cloud on the

table plane. An example of such a projection can be seen in Figure 5(f) (left). The corresponding grasp is a top grasp approaching the centre of mass of this projection. The wrist orientation of the hand is determined such that the vector between fingers and thumb is aligned with the minor eigenvector. A grasping point $G^{C_0} = \begin{bmatrix} zx & zy & z \end{bmatrix}$ in camera space C_0 is then formed with the x and y coordinates of the center of the segmentation mask as obtained during fixating on the object. The depth of this point, i.e, the z coordinate is computed from the vergence angle as read from the motor encoders. The goal is then to align the tool center point of the end effector with this grasping point. Using more sophisticated grasp planners that can deal with more complex scenarios is regarded as future work.

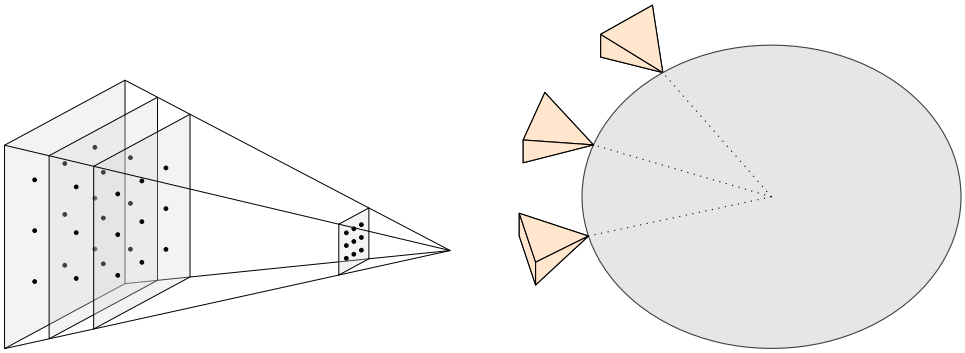
4.3. Object Localisation in Different Viewing Frames

The robotic head moves during the grasping process for focusing on different parts of the environment (different objects, the robotic hand and arm). In each of these views the object to be grasped should remain localized relative to the current camera frame C_k to accurately align the end effector with it. The new position of the object can be estimated given the new pose of the head, which is determined based on the motor reading and the forward kinematics of the head as depicted in Figure 1(b). However, we cannot completely rely on this estimate due to inaccuracies in the kinematic model and motor encoders. For this reason we perform a local refinement of the object position in the new view of the scene based on template matching.

A template is generated for each object when the head is fixated on it. Based on the segmentation mask in the foveal view as shown in Figure 5(d) and the calibration between the peripheral and foveal cameras, we generate a tight bounding box around the object in the peripheral view. Each time the head moves, the new position of this template can be estimated based on the old and new pose of the head. We create a window of possible positions of the object by growing this initial estimate by a fixed amount of pixels. We perform a sliding window comparison between all possible locations of the object template within this window and the object template. The location which returns the lowest mean square error is the one suggested as x and y coordinates of the object position in the new view.

4.4. Offline Calibration Process

In Section 2.2, we analysed the errors that got introduced by the different parts of the system. The group of systematic errors can be minimised by



(a) Movement pattern of the end effector for offline calibration.

(b) Three viewing directions of a camera rotated around one axis.

Figure 6: Offline Calibration procedure (video available at `http://www.youtube.com/watch?v=dEytfUgmcfA`).

offline calibration. In the following, we will introduce our method for stereo and hand-eye calibration as well as the calibration of the kinematic chain of the robotic head.

4.4.1. Stereo Calibration

One of the most commonly used methods for finding the transformation between two camera coordinate systems is the use of a checkerboard which is observed by two cameras (or the same camera before and after moving) [34]. The checkerboard defines its own coordinate system in which the corners of the squares are the set of 3D points Q^A in the arm coordinate system A as in Section 2.2.1. The detection of these corners in the left and right images gives us the corresponding pixel coordinates P from which we can solve for the internal and external camera parameters. Given these, we can obtain the transformation between the left and right camera coordinate frame.

We used a modified version of this method. Instead of a checkerboard pattern, we used a small LED rigidly attached to the end effector of the robotic arm, which we can detect in the image with subpixel precision. Because of the accuracy and repeatability of the KUKA arm (< 0.3mm), we can move the LED to a number of places for which we know the exact position in arm space, which allows us to obtain the transformation between arm and camera space.

This method has several advantages over the use of a traditional checkerboard pattern:

- Instead of an arbitrary checkerboard coordinate system as intermediate coordinate system, we can use the arm coordinate system. In this way we are obtaining the hand-eye calibration for free at the same time that we are performing the stereo calibration.

- In the checkerboard calibration method, the checkerboard ought to be fully visible in the two calibrating cameras for every calibration pose. This may be difficult when the two camera poses are not similar or their field of view is small. With our approach, it is not necessary to use exactly the same end effector positions for calibrating the two cameras since all points are defined in the static arm coordinate system that is always valid independently of camera pose.

- For these same reasons, we found empirically that this approach makes it possible to choose a pattern that offers a better calibration performance. For example, by using a set of calibration points that is uniformly distributed in camera space (as opposed to world space, which is the case for checkerboard patterns), it is possible to obtain a better characterization of the lenses distortion parameters.

The main building block for this system is a visual servoing loop that allows us to bring the LED to several predefined positions in camera space. In this loop, we want to control the position of the LED (3 DOF) using only its projection in the image (2 DOF). Therefore, we introduce an additional constraint by limiting the movement of the LED to a predefined plane in arm space.

We define S_0^A as a point on this plane and S^A as its normal vector. Additionally, we need a rough estimate of the arm to camera coordinate transformation $\boldsymbol{A}_A^C$, and of the camera matrix $\boldsymbol{C}$. They are defined as follows:

$$\boldsymbol{A}_A^C = \left[\boldsymbol{R}_A^C | \boldsymbol{t}_A^C\right], \boldsymbol{C} = \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{8}$$

We can then find the plane in camera space:

$$S_0^C = \boldsymbol{R}_A^C S_0^A + \boldsymbol{t}_A^C \qquad S^C = \boldsymbol{R}_A^C S^A \tag{9}$$

The process of moving the LED to a position in the image is then as follows: we define a target point in the image $P_t = (u_t, v_t)$ (specified in

pixels) where we want to bring the LED. For each visual servoing iteration we detect the current position $P_c = (u_c, u_c)$ (again in pixels) of the LED in the image. We can then use the camera matrix to convert these points to homogeneous camera coordinates:

$$P_t^C = \boldsymbol{C}^{-1} \begin{bmatrix} x_t & y_t & 1 \end{bmatrix}^{\mathrm{T}} = \begin{bmatrix} \frac{1}{f} x_t & \frac{1}{f} y_t & 1 \end{bmatrix}^{\mathrm{T}} \tag{10}$$

$$P_c^C = \boldsymbol{C}^{-1} \begin{bmatrix} x_c & y_c & 1 \end{bmatrix}^{\mathrm{T}} = \begin{bmatrix} \frac{1}{f} x_c & \frac{1}{f} y_c & 1 \end{bmatrix}^{\mathrm{T}} \tag{11}$$

Then, we can project these homogeneous points into the plane defined by S_0^C and S^C. A point Q is in that plane if $Q \cdot S^C = S_0^C \cdot S^C$. Therefore, the projection of the homogeneous points P_t^C and P_c^C into the plane can be found as

$$Q_t^C = \frac{S_0^C \cdot S^C}{P_t^C \cdot S^C} P_t^C \qquad Q_c^C = \frac{S_0^C \cdot S^C}{P_c^C \cdot S^C} P_c^C \tag{12}$$

From these, we can obtain the vector

$$d^C = Q_t^C - Q_c^C \tag{13}$$

which is the displacement from the current to the target LED positions in camera space, and then

$$d^A = \boldsymbol{R}_A^{C^{-1}} d^C \tag{14}$$

which is the correspondent displacement in arm space.

We can easily see that the displacement obtained in arm space is within the given plane since

$$d^C \cdot S^C = (Q_t^C - Q_c^C) \cdot S^C = Q_t^C \cdot S^C - Q_c^C \cdot S^C = 0 \tag{15}$$

$$d^A \cdot S^A = d^{A^{\mathrm{T}}} S^A = (\mathbf{R}^{-1} d^C)^{\mathrm{T}} (\mathbf{R}^{-1} S^C) = d^{C^{\mathrm{T}}} \mathbf{R} \mathbf{R}^{-1} s^C = d^C \cdot S^C = 0 \tag{16}$$

We can then use this displacement to obtain a simple proportional control law

$$\boldsymbol{u} = k d^A \tag{17}$$

where $\boldsymbol{u}$ is the velocity of the end effector and k is a constant gain factor.

Once we have the system which allows us to bring the LED to a certain position in the image we can start generating our calibration pattern. First, we bring the LED to the center of the image in two parallel planes, which are located at different distances from the camera, and record their positions C_0^A and C_1^A in arm space. From this, we can obtain the vector $S^A = C_0^A - C_1^A$

Algorithm 1: Pseudo Code for Defining the Calibration Pattern

Data: Number of points n on each depth plane, number of depth planes m
Result: Set of Correspondences $\left[Q_i^A, P_i\right]$ with $(0 < i \leq m \cdot n)$
begin
  `// Initialising the principal axis of the camera in arm space`
  `// VisualServoing(P,S) is a function that brings the LED to`
  `// the point P in image space within the plane S in arm space`
  $\mathbf{S}_0$ = Some plane at a distance d_0 from the camera
  C_0^A = `VisualServoing(`$(0,0), \mathbf{S}_0$`)`
  $\mathbf{S}_1$ = Some plane at a distance d_1 from the camera
  C_1^A = `VisualServoing(`$(0,0), \mathbf{S}_1$`)`
  $d = (C_1^A - C_0^A)/(m-1)$
  $i = 0$
  foreach $l \in [0 \ldots m-1]$ **do**
    $\mathbf{S}$ = plane with normal d which contains $C_0^A + ld$
    foreach P_k *with* $(0 < k \leq n)$ **do**
      Q_i^A = `VisualServoing(`$(x_k, x_y), \mathbf{S}$`)`
      $i++$
    end
  end
end

which is parallel to the principal axis of the camera. Any plane perpendicular to this vector will be parallel to the image plane. We can then bring the LED to some fixed positions along these planes, thus generating points which are uniformly distributed both in the image and in depth (by using planes which are separated by a constant distance). The generated pattern looks like the one shown in Figure 6(a). Algorithm 1 shows the method in more detail. In our system, we used 6x6 points rectangular patterns in 6 different depths, for a total of 216 calibration points. With this, we achieved an average reprojection error of 0.1 pixels.

4.4.2. Head-Eye Calibration

For a static camera setup, the calibration process would be completed here. However, our vision system can move to fixate on the objects we manipulate. Due to inaccuracies in the kinematic model of the head, we cannot obtain the exact transformation between the camera coordinate system before and after moving a certain joint from the motor encoders.

In Section 2.2.2, we modelled the transformation error that arises from the misalignment of the real and ideal coordinate frame of a joint. In our method,

we are minimising this error by finding the true position and orientation of the real coordinate frame as follows:

1. Choose two different positions of the joint, that are far enough apart to be significant, but with an overlapping viewing area that is still reachable for the robotic arm.
2. For each of these two positions, perform the static calibration process as described above, so that we obtain the transformation between the arm coordinate system and each of the camera coordinate systems.
3. Find the transformation between the camera coordinate systems in the two previously chosen joint configurations. This transformation is the result of rotating the joint around some roughly known axis (it is not precisely known because of mechanical inaccuracies), with a roughly known angle from the motor encoders. From the computed transformation, we can then more exactly determine this axis, center and angle of rotation.

This is illustrated in Figure 6(b).

In our case, the results showed that while the magnitude of the angles of rotation differed significantly from what could be obtained from the kinematic chain, the orientation and position of these axes were quite precise in the specifications. To avoid overly complicating the model, we decided to correct only the actual angles α in Equation 6 , except for the vergence joint. For this joint, the orientation was also corrected, since here small errors have a large impact in the accuracy of depth estimation. Therefore, we minimised the discrepancy between the true and estimated essential matrix in Equations 3 and 5 respectively.

4.5. Virtual Visual Servoing

As mentioned in Sections 1 and 3.3, our system uses Virtual Visual Servoing to refine the position of the arm and hand in camera space provided by the calibration system. In Figure 8, the difference between the estimated arm pose and the corrected one is visualised. In this section we will formalize the problem and explain how we solved it.

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

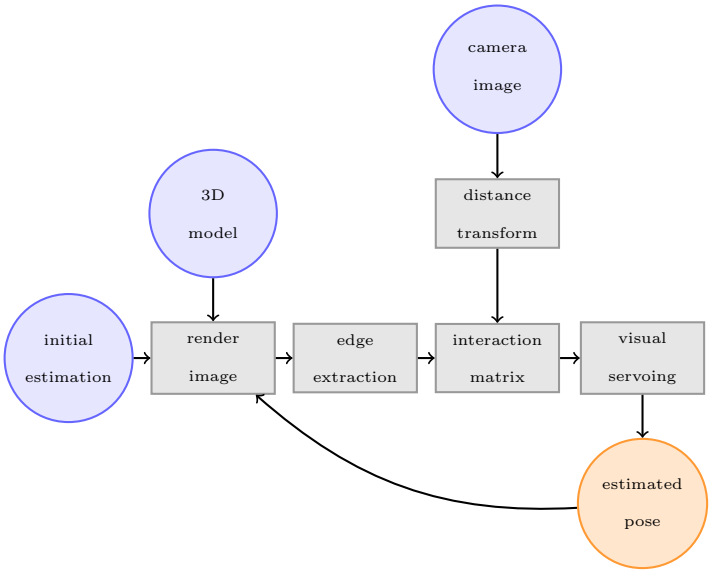
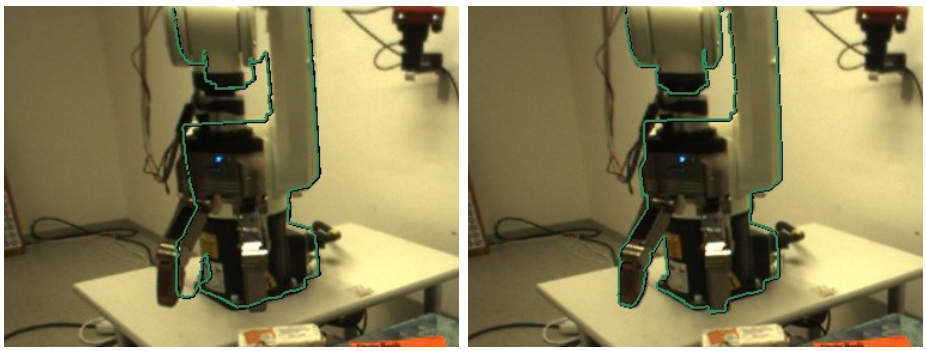


Figure 7: Outline of the proposed model-based tracking system based on Virtual Visual Servoing.



(a) (b)

Figure 8: Comparison of robot localization with (right) and without (left) the Virtual Visual Servoing correction.

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

$$\mathbf{M} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & T_X \\ r_{21} & r_{22} & r_{23} & T_Y \\ r_{31} & r_{32} & r_{33} & T_Z \end{bmatrix} \tag{18}$$

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

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \mathbf{R}\left[\mathbf{Q} - \mathbf{t}\right] \quad \text{with} \quad [wx,\ wy,\ w] = \mathbf{P} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \tag{19}$$

where $\mathbf{P}$ represents the internal camera parameters matrix including focal length and aspect ratio of the pixels, w is the scaling factor, $\mathbf{R}$ and $\mathbf{t}$ represent the rotation matrix and translation vector, respectively.

Our approach to pose estimation and tracking is based on virtual visual servoing where a rendered model of the robot parts is aligned with the current image of their real counterparts. The outline of the system is presented in Fig. 7. In order to achieve the alignment, we can either control the position of the part to bring it to the desired pose or move the *virtual* camera so that the *virtual* image corresponds to the current camera image, denoted as *real* camera image in the rest of the paper. Using the latter approach has the problem that the local effect of a small change in orientation of the camera is very similar to a large change in its position, which leads to convergence problems. Therefore, in this paper, we adopt the first approach where we render synthetic images by incrementally changing the pose of the tracked part. In the first iteration, the position is given based on the forward kinematics. Then, we extract visual features from the rendered image, and compare them with the features extracted from the current camera image. The details about the features that are extracted are given in Section 4.5.2.

Based on the extracted features, we define an error vector

$$\boldsymbol{s}(t) = [d_1, d_2, \ldots, d_n]^{\mathrm{T}} \tag{20}$$

between the desired values for the features and the current ones. Based on $\boldsymbol{s}$, we can estimate the incremental change in pose in each iteration $\boldsymbol{e}(s)$ following the classical image based servoing control loop. This process continues

until the difference vector $\boldsymbol{s}$ is smaller than a certain threshold. Each of the steps is explained in more detail in the following subsections. Our current implementation and experimental evaluation is performed for the Kuka R850 arm and Schunk Dexterous hand, shown in Fig. 9

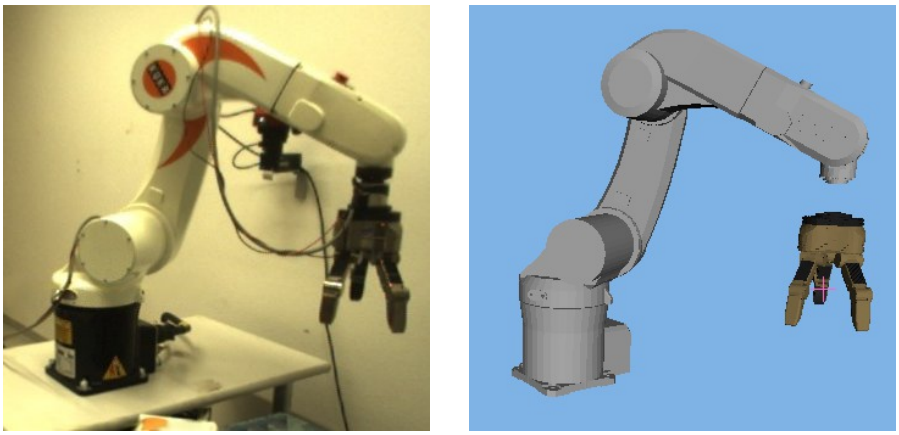


Figure 9: The Kuka R850 arm and Schunk Dexterous Hand in real images and CAD models.

4.5.1. Virtual image generation

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

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

One of the most common CAD formats for objects such as robotic hands are Inventor files. There are a number of rendering engines which can deal with such models, but none of them had the performance and flexibility that we needed. For that we developed a new scenegraph engine, specific for this application, which focused on rendering offline images at a high speed. It was developed directly over OpenGL. We can obtain about 1000 frames per second with this engine. At the moment the speed of the rendering engine does not represent a bottleneck to our system.

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

4.5.2. Features

The virtual and the real image of the robot parts ought to be compared in terms of visual features. These features should be fast to compute, because this comparison will be performed as many times per frame as iterations are needed by the virtual visual servoing for locating the robot. They should also be robust towards non-textured models.

Edge-based features fulfill these requirements. In particular chamfer distances [35] modified to include the alignment of edge orientations [36] are well suited for matching shapes in cluttered scenes, and are used in recent systems for object localization [37]. This feature is similar in spirit but more general than other ones used in the field of Virtual Visual Servoing. Comport et al. [38] computes distances between real points and virtual lines/ellipsis instead of virtual points. Klose et al. [39], Drummond and Cipolla [40] search for real edges only in the perpendicular direction of the virtual edge.

The mathematic formulation of our features is the following. After performing a Canny edge detection on real image I_u and virtual image I_v we obtain a set of edge points $\mathcal{U}, \mathcal{V}$ with their correspondent edge orientations $\mathcal{O}_\mathcal{U}, \mathcal{O}_\mathcal{V}$ (from the horizontal and vertical Sobel filter). The sets $\mathcal{U}, \mathcal{V}$ are split into overlapping subsets $\mathcal{U}_i, \mathcal{V}_i$ according to their orientations:

$$o_u \pmod{\pi} \in [\frac{2\pi i}{16}, \frac{2\pi(i+1)}{16}] \Rightarrow u \in \mathcal{U}_i \tag{21}$$

Then for each channel $\mathcal{U}_i$ we compute the distance transform [41] which will allow us to perform multiple distance computations for the same real set $\mathcal{U}$ in linear time with the number of points in each new set $\mathcal{V}$. Based on the distance transform we obtain our final distance vector $\boldsymbol{s} = \{d_{cham}(v,\mathcal{U})\} = [d_1, d_2 \cdots d_n]^{\mathrm{T}}$ composed by distance estimations for each edge point from our virtual image I_v.

$$d_{transf}(p) = \min_{u \in \mathcal{U}_i} ||p - u||, \qquad p \in I_u \tag{22}$$

$$d_{cham}(v,\mathcal{U}) = d_{transf}(v), \qquad v \in \mathcal{V}_i, u \in \mathcal{U}_i \tag{23}$$

$$\boldsymbol{s} = \{d_{cham}(v,\mathcal{U})\}, \qquad d_{cham}(v,\mathcal{U}) < \delta \tag{24}$$

The points $v \in \mathcal{V}$ whose distance is higher than an empirically estimated threshold δ are considered outliers and dropped from $\boldsymbol{s}$.

4.5.3. Pose correction

Once the features have been extracted, we can use a classical visual servoing approach to calculate the correction to the pose, [42]. There are two different approaches to vision-based control: *Position-based* control uses image data to extract a series of 3D features, and control is performed in the 3D Cartesian space. In *image-based* control, the image features are directly used to control the robot motion. In our case, since the features we are using are distances between edges in an image for which we have no depth information in the real image, we are using an image-based approach.

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

As discussed before, $\boldsymbol{s}(t)$ is a vector of feature values measured in the image, composed by distances between edges in the real and synthetic images I_u, I_v. Therefore $\dot{\boldsymbol{s}}(t)$ will be the rate of change of these distances with time.

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

$$\dot{\boldsymbol{r}}(t) = \left[T_x, T_y, T_z, \omega_x, \omega_y, \omega_z\right]^{\mathrm{T}} \tag{25}$$

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

$$\dot{\boldsymbol{s}} = \boldsymbol{J}\dot{\boldsymbol{r}} \tag{26}$$

where

$$\boldsymbol{J} = \left[\frac{\partial \boldsymbol{s}}{\partial \boldsymbol{r}}\right] = \begin{bmatrix} \frac{\partial d_1}{\partial T_x} & \frac{\partial d_1}{\partial T_y} & \frac{\partial d_1}{\partial T_z} & \frac{\partial d_1}{\partial \omega_x} & \frac{\partial d_1}{\partial \omega_y} & \frac{\partial d_1}{\partial \omega_z} \\ \frac{\partial d_2}{\partial T_x} & \frac{\partial d_2}{\partial T_y} & \frac{\partial d_2}{\partial T_z} & \frac{\partial d_2}{\partial \omega_x} & \frac{\partial d_2}{\partial \omega_y} & \frac{\partial d_2}{\partial \omega_z} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \frac{\partial d_n}{\partial T_x} & \frac{\partial d_n}{\partial T_y} & \frac{\partial d_n}{\partial T_z} & \frac{\partial d_n}{\partial \omega_x} & \frac{\partial d_n}{\partial \omega_y} & \frac{\partial d_n}{\partial \omega_z} \end{bmatrix} \tag{27}$$

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

below.

However, to be able to correct our pose estimation, we need the opposite: we need to compute $\dot{\boldsymbol{r}}(t)$ given $\dot{\boldsymbol{s}}(t)$.

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

$$\dot{\boldsymbol{r}} = \boldsymbol{J}^{+}\dot{\boldsymbol{s}} \tag{28}$$

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

$$\boldsymbol{J}^{+} = (\boldsymbol{J}^{\mathrm{T}}\boldsymbol{J})^{-1}\boldsymbol{J}^{\mathrm{T}}. \tag{29}$$

The goal for our task is to have all the edges in our synthetic image match edges in the real image, so the target value for each of the features is 0. Then, we can define the error function as

$$\boldsymbol{e}(\boldsymbol{s}) = \boldsymbol{0} - \dot{\boldsymbol{s}} \tag{30}$$

which leads us to the simple proportional control law:

$$\dot{\boldsymbol{r}} = -K\boldsymbol{J}^{+}\boldsymbol{s} \tag{31}$$

where K is the gain parameter.

4.5.4. Estimation of the jacobian

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

In our case, the features in $\boldsymbol{s}$ are the distances from the edges of the synthetic image to the closest edge in the real image. Therefore, we numerically approximate the derivative by calculating how a small change in the relevant direction affects the value of the feature.

As we said before, while rendering the model we obtained a depth map. From this depth map, it is possible to obtain the 3D point corresponding to each of the edges. Each model point v in I_v is a projection of its corresponding 3D point in the model $\boldsymbol{v_m}$. The derivative of the distance described before $d_{cham}(v,\mathcal{U})$, can be calculated for a model point $\boldsymbol{v_m}$ with respect to T_x by applying a small displacement and projecting it into the image:

$$\frac{d_{cham}\big(\boldsymbol{PM}(\boldsymbol{v_m}+\epsilon\boldsymbol{x}),\mathcal{U}\big)-D\big(\boldsymbol{PM}\boldsymbol{v_m},\mathcal{U}\big)}{\epsilon} \tag{32}$$

where ϵ is an arbitrary small number and $\boldsymbol{x}$ is a unitary vector in the x direction. $\boldsymbol{P}$ and $\boldsymbol{M}$ are the projection and modelview matrices, as defined in Section 4.5.1. A similar process is applied to each of the motion components.

4.6. Object grasping

In this section we will combine the grasping point computed in Section 4.2 and the arm pose calculated in Section 4.5 in order to move the arm so that it can grasp the object.

After finding the grasping position as a point G^{C_0} in camera space, we move the head to a position where the whole arm is visible, while keeping the object in the field of view. Then, the object position (x, y) is found in this new viewpoint using the method in Section 4.3. We assume that the z coordinate did not change with the gaze shift. Therefore, we use the one calculated previously in G^{C_0}. The grasping point in camera space for the current viewpoint is then $G^{C_1} = \begin{bmatrix} zx & zy & z \end{bmatrix}$.

After this, virtual visual servoing allows us to obtain the transformation $A^A_{C_1}$ that converts points from camera to arm space, so we can obtain the grasping point in arm space as $G^A = A^A_{C_1} G^{C_1}$. To account for small errors in the measurements, we do not move the arm there directly, but take it first to a position which is a few centimeters (15 cm in our experiments) above G^A.

Then, the final step is to bring the arm down so that the object lies between the fingers of the hand. We do this using a simple visual servoing loop. The movement to be performed is purely vertical. Therefore, we only use a single visual feature to control that degree of freedom: the vertical distance between the arm and the grasping point in the image which will be roughly aligned with the vertical axis of the arm. In each iteration, we measure the vertical distance d between the arm and the grasping point in the image, and use this as input for a simple proportional control law:

$$\dot{r}_y = -kd \tag{33}$$

where $\dot{r}_y$ is the vertical velocity of the arm and k is a gain factor.

5. Experiments

5.1. Robustness of Virtual Visual Servoing

In order to evaluate the performance of the virtual visual servoing, we needed a setup where ground truth data was available, so that the error both for the input (edge detection and initial estimation) and the output (estimated pose) is known. Lacking such a setup, we decided to conduct the experiments using synthetic images as input. These images were generated using the same 3D model and rendering system that we use in the visual servoing loop.

The process used in these experiments was as follows:

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

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

- Translational error. We evaluated which range of errors in the translational component of the initial pose estimation allows the system to converge. To that effect, we added, in each run, a translational error of known magnitude and random direction.
- Rotational error. We also evaluated the range of errors in the rotational component of the initial pose estimation for which the system converges to the correct result. For each run, we added a rotational error of known magnitude and random direction.

- Error in edge detection. To simulate the effects of wrongly detected edges in the performance of the system, we added random errors to the edge detection of the input synthetic image. In each run, the detection of a number of pixels was shifted a random amount. An example of the result can be seen in Fig. 10.

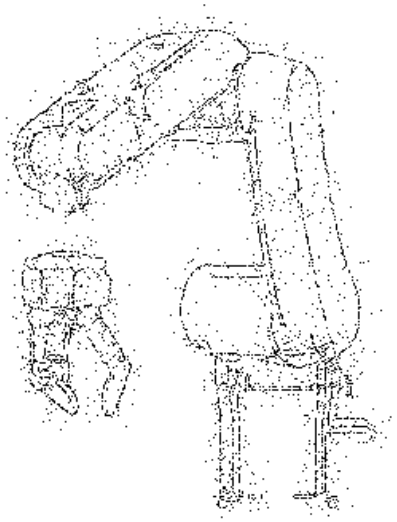


Figure 10: Edge detection with artificially introduced error (in 30% of the pixels).

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

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

- The threshold for deciding that the algorithm had converged to the correct pose was 10 mm in translation and 0.5° in rotation.
- The number of iterations after which the run was considered a failure was set to 200.

- For the experiments that evaluate robustness with respect to edge detection, a translation error of 100 mm and a rotation error of 10° was used for the initial pose estimation.

Figure 11 shows the result of increasing the translational component of the error in the estimated initial pose. As we can see, the failure rate is close to 0 for distances below 100 mm, and then starts increasing linearly. This is probably due to 100 mm being in the same order of magnitude as the distance between different edges of the robot which have the same orientation, so the system gets easily lost, trying to follow the wrong edges. We can also observe an increase in the iterations needed for reaching the result.

In Figure 12 we can observe a similar behaviour for the tolerance to errors in the rotational component of the estimated initial pose. Here, the threshold is around 10° and the reason is probably the same as before: this is the minimum rotation that bring edges to the position of other edges.

With respect to the error in edge detection, we can see in Figure 13 that when less than 50% of the pixels are wrongly detected, the system performs almost as well as with no error, and after that the performance quickly degrades. It is also significant that for the runs that converge successfully, the number of iterations is almost independent of the errors in edge detection.

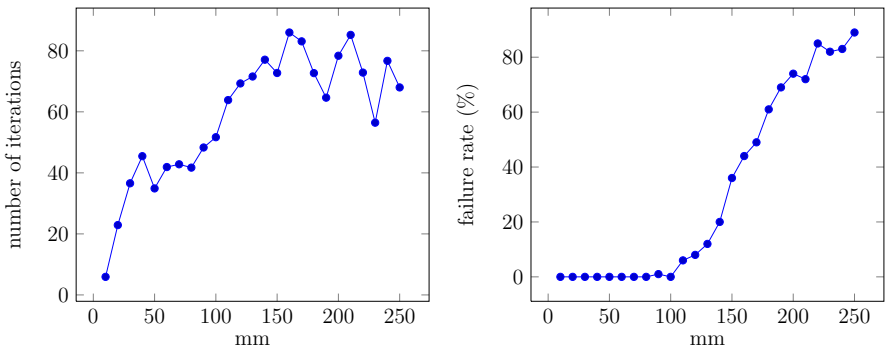


Figure 11: Effects of translational error

5.2. Qualitative Experiments on the Real System

For our experiments with the real robot, we set up a table-top scenario with two randomly placed objects, as can be seen in the last picture in Figure 14. The head was initially looking towards the table, where it could

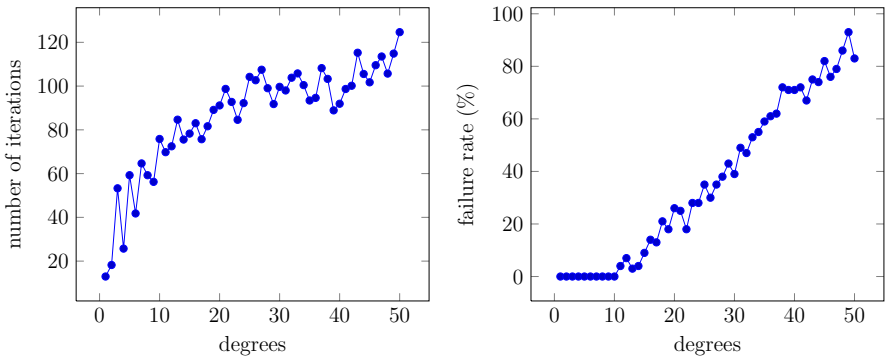


Figure 12: Effects of rotational error

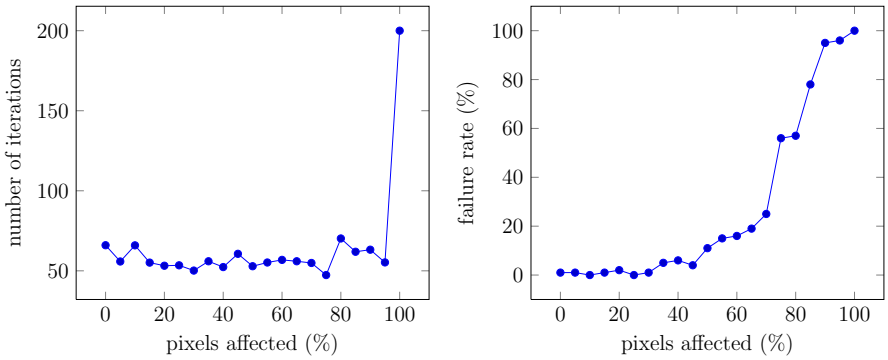


Figure 13: Effects of errors in edge detection

see the objects and find them in the saliency map of the attention system. However, the arm was not fully visible.

We ran the system several times with this setup. In most of these runs, the virtual visual servoing loop did not converge because it could only see a small part of the arm. Therefore, we decided to run virtual visual servoing only after the object is found and after shifting the gaze away from it and towards the arm. With this change, even if the initial estimation for the pose of the arm was significantly different, the pose estimated through visual servoing quickly converged. In Figure 14 we can see, outlined in green, the estimated pose for the arm and hand over the real image.

The segmentation of the object and detection of the grasping point worked well. When running virtual visual servoing with the full arm in view, the hand could be aligned with the object with high accuracy and grasping was

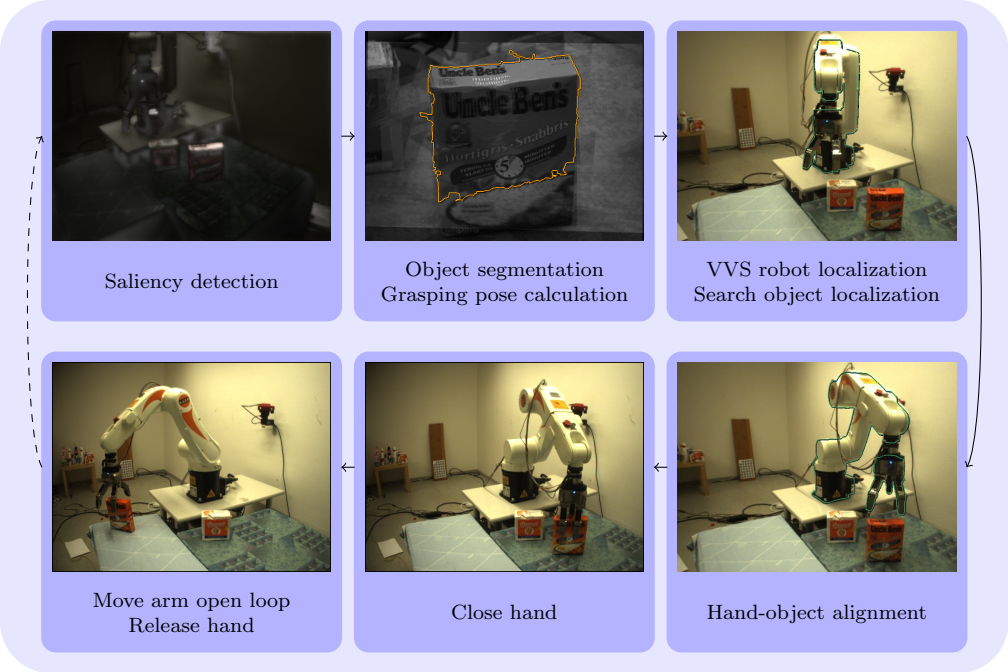


Figure 14: Grasping system diagram. A video with the whole sequence can be found at `http://www.youtube.com/watch?v=e-03Y8_cgPw`

performed successfully in most of the runs.

However, even if they did not affect the performance of the system, there are some problems that arise with the use of virtual visual servoing in real images. For example, as we can see in the upper-right picture of Figure 14, the upper edge of the reconstructed outline does not match exactly with the upper edge of the arm. There are two main reasons for this. First, the illumination of the scene can lead to some edges disappearing, because of the low contrast. This is usually not a problem, because the correct matching of other edges will compensate for this. But in this case, there is a second problem: because of the orientation of the arm with respect to the camera, the outer silhouette has really few edges. As we can see in the picture below that, once the pose of the arm changes, the system can recover and show the correct pose again. As future work, we are considering the possibility of using not only the outer contour, but also try to match some of the internal edges of the model. While this can lead to a less robust system, since outer

edges are more likely to appear as edges in the real image, it can increase performance where the number of edges is low.

6. Conclusions

Grasping and manipulation of objects is a necessary capability of any robot performing realistic tasks in a natural environment. The solutions require a systems approach since the objects need to be detected and modelled prior to an agent acting upon them. Although many solutions for known objects have been proposed in the literature, dealing with unknown objects stands as an open problem.

Our research deals with this problem by integrating the areas of active vision and sensor based control where visual servoing methods represent important building blocks. In this paper, we have presented several ways in which these methods can add robustness and help minimizing the errors inherent to any real system.

First, we showed how visual servoing can be used to automate the offline calibration process. The robot can, without human intervention, generate a pattern to collect data that can then be used to calibrate the cameras and the transformations between the cameras and the arm.

Then, we demonstrated a virtual visual servoing approach to continuously correct the spatial relation between the arm and the cameras. Though in its current state the system works well and is useful in the context of the system, some ideas for future work arise from the experiments we performed. Using the edges as the only feature in the visual servoing loop works well with objects which have a complex outline, such as our arm. However, for other kinds of object, or even some viewpoints of the arm, having a wider range of features might help. These features may include adding textures to the models, or using 3D information obtained via stereo or structured light cameras.

Finally, the last step that brings the arm to the place where the object can actually be grasped also uses visual information to move the arm down to the position where the hand can be closed to grasp the object. There is also room for future improvements here, such as integrating the system with a more complex grasp planner, which could determine the optimal points where the fingers should contact the object. Then, visual servoing could be used to visually bring the fingers to the correct position.

Acknowledgement

This work is supported by the EU through the project GRASP, IST-FP7-IP-215821, Swedish Foundation for Strategic Research and UKF-Unity through Knowledge Fund.

References

[1] J. Glover, D. Rus, N. Roy, Probabilistic Models of Object Geometry for Grasp Planning, in: IEEE International Conference on Robotics and Automation, Pasadena, CA, USA, 2008.

[2] K. Huebner, K. Welke, M. Przybylski, N. Vahrenkamp, T. Asfour, D. Kragic, R. Dillmann, Grasping Known Objects with Humanoid Robots: A Box-based Approach, in: International Conference on Advanced Robotics, 1–6, 2009.

[3] B. Rasolzadeh, M. Björkman, K. Huebner, D. Kragic, An Active Vision System for Detecting, Fixating and Manipulating Objects in Real World, Int. J. of Robotics Research To appear.

[4] P. Azad, T. Asfour, R. Dillmann, Stereo-based 6D Object Localization for Grasping with Humanoid Robot Systems, in: IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS), 919–924, 2007.

[5] J. Bohg, D. Kragic, Learning Grasping Points with Shape Context, Robotics and Autonomous Systems In Press.

[6] A. Saxena, J. Driemeyer, J. Kearns, A. Y. Ng, Robotic Grasping of Novel Objects, Neural Information Processing Systems 19 (2007) 1209–1216.

[7] J. Speth, A. Morales, P. J. Sanz, Vision-Based Grasp Planning of 3D Objects by Extending 2D Contour Based Algorithms, in: IEEE/RSJ International Conference on Intelligent Robots and Systems, 2240–2245, 2008.

[8] M. Stark, P. Lies, M. Zillich, J. Wyatt, B. Schiele, Functional Object Class Detection Based on Learned Affordance Cues, in: 6th International Conference on Computer Vision Systems, vol. 5008 of *LNAI*, Springer-Verlag, 435–444, 2008.

[9] K. Huebner, D. Kragic, Selection of Robot Pre-Grasps using Box-Based Shape Approximation, in: IEEE/RSJ International Conference on Intelligent Robots and Systems., 1765–1770, 2008.

[10] M. Richtsfeld, M. Vincze, Grasping of Unknown Objects from a Table Top, in: ECCV Workshop on 'Vision in Action: Efficient strategies for cognitive agents in complex environments', Marseille, France, 2008.

[11] D. Aarno, J. Sommerfeld, D. Kragic, N. Pugeault, S. Kalkan, F. Wörgötter, D. Kraft, N. Krüger, Early Reactive Grasping with Second Order 3D Feature Relations, in: ICRA Workshop: From Features to Actions, 319–325, 2007.

[12] N. Bergström, J. Bohg, D. Kragic, Integration of Visual Cues for Robotic Grasping, in: Computer Vision Systems, vol. 5815 of *Lecture Notes in Computer Science*, Springer Berlin / Heidelberg, 245–254, 2009.

[13] D. Kragic, H. I. Christensen, Cue Integration for Visual Servoing, IEEE Transactions on Robotics and Automation 17 (1) (2001) 18–27.

[14] T. Asfour, K. Welke, P. Azad, A. Ude, R. Dillmann, The Karlsruhe Humanoid Head, in: IEEE/RAS International Conference on Humanoid Robots (Humanoids), Daejeon, Korea, 2008.

[15] KUKA, KR 5 sixx R850, `www.kuka-robotics.com`, Last Visited Dec'10.

[16] SCHUNK, SDH, `www.schunk.com`, Last visited Dec'10.

[17] X. Gratal, J. Bohg, M. Björkman, D. Kragic, Scene Representation and Object Grasping Using Active Vision, in: IROS'10 Workshop on Defining and Solving Realistic Perception Problems in Personal Robotics, 2010.

[18] A. W. Khan, C. Wuyi, Systematic Geometric Error Modelling for Workspace Volumetric Calibration of a 5-axis Turbine Blade Grinding Machine, Chinese Journal of Aeronautics (2010) 604–615.

[19] J. Felip, A. Morales, Robust sensor-based grasp primitive for a three-finger robot hand, in: Proceedings of the 2009 IEEE/RSJ international conference on Intelligent robots and systems, IROS'09, 1811–1816, 2009.

[20] K. Hsiao, S. Chitta, M. Ciocarlie, E. G. Jones, Contact-Reactive Grasping of Objects with Partial Shape Information, in: IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS), Taipei, Taiwan, 1228–1235, 2010.

[21] A. Ude, D. Omrcen, G. Cheng, Making Object Learning and Recognition an Active Process, Int. J. of Humanoid Robotics 5 (2) (2008) 267–286.

[22] N. Vahrenkamp, S. Wieland, P. Azad, D. Gonzalez, T. Asfour, R. Dillmann, Visual Servoing for Humanoid Grasping and Manipulation Tasks, in: IEEE/RAS Int. Conf. on Humanoid Robots (Humanoids), 406–412, 2008.

[23] A. Ude, E. Oztop, Active 3-D Vision on a Humanoid Head, in: Int. Conf.on Advanced Robotics (ICAR), Munich, Germany, 2009.

[24] K. Welke, M. Przybylski, T. Asfour, R. Dillmann, Kinematic Calibration for Saccadic Eye Movements, Tech. Rep., Institute for Anthropomatics, Universität Karlsruhe, 2008.

[25] V. Lepetit, J. Pilet, P. Fua, Point matching as a classification problem for fast and robust object pose estimation, in: CVPR, II: 244–250, 2004.

[26] T. Drummond, R. Cipolla, Real-Time Visual Tracking of Complex Structures, IEEE Trans. PAMI 24 (7) (2002) 932–946.

[27] V. Kyrki, D. Kragic, Integration of model-based and model-free cues for visual object tracking in 3D, in: IEEE International Conference on Robotics and Automation, ICRA'05, 1566–1572, 2005.

[28] D. Kragic, V. Kyrki, Initialization and System Modeling in 3-D Pose Tracking, in: In IEEE International Conference on Pattern Recognition 2006. ICPR'06, Hong Kong, 643–646, 2006.

[29] A. I. Comport, D. Kragic, E. Marchand, F. Chaumette, Robust Real-Time Visual Tracking: Comparison, Theoretical Analysis and Performance Evaluation, 2841 – 2846, 2005.

[30] J. Romero, H. Kjellström, D. Kragic, Hands in action: real-time 3D reconstruction of hands in interaction with objects, in: ICRA, IEEE, 458–463, 2010.

[31] B. León, S. Ulbrich, R. Diankov, G. Puche, M. Przybylski, A. Morales, T. Asfour, S. Moisio, J. Bohg, J. Kuffner, R. Dillmann, OpenGRASP: A Toolkit for Robot Grasping Simulation, in: SIMPAR '10: Proceedings of the 2nd International Conference on Simulation, Modeling, and Programming for Autonomous Robots, 2010.

[32] M. Björkman, D. Kragic, Active 3D Scene Segmentation and Detection of Unknown Objects, in: IEEE International Conference on Robotics and Automation, 2010.

[33] M. Björkman, D. Kragic, Active 3D Segmentation through Fixation of Previously Unseen Objects, in: British Machine Vision Conference, to appear, 2010.

[34] Z. Zhang, A Flexible New Technique for Camera Calibration, IEEE Transactions on Pattern Analysis and Machine Intelligence 22 (2000) 1330–1334, ISSN 0162-8828.

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

[36] D. M. Gavrila, Multi-Feature Hierarchical Template Matching Using Distance Transforms, in: ICPR, Vol I: 439–444, 1998.

[37] M. Liu, O. Tuzel, A. Veeraraghavan, R. Chellappa, Fast directional chamfer matching, in: CVPR, IEEE, 1696–1703, 2010.

[38] A. I. Comport, É. Marchand, M. Pressigout, F. Chaumette, Real-Time Markerless Tracking for Augmented Reality: The Virtual Visual Servoing Framework, IEEE Trans. Vis. Comput. Graph 12 (4) (2006) 615–628.

[39] S. Klose, J. Wang, M. Achtelik, G. Panin, F. Holzapfel, A. Knoll, Markerless, Vision-Assisted Flight Control of a Quadrocopter, in: IROS, 2010.

[40] T. Drummond, R. Cipolla, Real-Time Visual Tracking of Complex Structures, IEEE Trans. Pattern Anal. Mach. Intell 24 (7) (2002) 932–946.

[41] G. Borgefors, Distance Transformations in Digital Images, Computer Vision, Graphics and Image Processing 34 (1986) 344–371.

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

Task Modeling in Imitation Learning using Latent Variable Models

Carl Henrik Ek, Dan Song, Kai Huebner and Danica Kragic

***Abstract*— An important challenge in robotic research is learning and reasoning about different manipulation tasks from scene observations.**

In this paper we present a probabilistic model capable of modeling several different types of input sources within the same model. Our model is capable to infer the task using only partial observations. Further, our framework allows the robot, given partial knowledge of the scene, to reason about what information streams to acquire in order to disambiguate the state-space the most.

We present results for task classification within and also reason about different features discriminative power for different classes of tasks.

I. Introduction

A major challenge in robot grasping is to automatically plan a grasp on an object that affords a specific task. Though proved to be an effective approach, learning by imitation [1], presents significant challenges to the robot sensory system. To observe a human demonstration, robots need to obtain the state of the entire scene, not only the objects, but also the human actions [2]. The problem becomes even harder when the goal is to perceive these features through vision systems [3]: human hands have many fingers that are often blocked by the grasped object, and objects are also mostly occluded by the hands. Understanding the scene from this huge range of noisy and uncertain sensory data is a formidable challenge, both in terms of computational resources and real-time applications. On the side of the motor system, another challenge in imitation learning is how to map a human grasp to a robot grasp, particularly when their hands are very different.

These challenges are not new in the field of robotics. To cope with the uncertainty in the sensorimotor systems, [4] proposed a coherent control, trajectory optimization, and action planning architecture. They applied the probabilistic inference-based methods and the dynamic Bayesian networks to integrate across all levels of representations. Another work [5] addresses the challenges in the robotics application with high degrees of freedom. They identified, from high-dimensional movement data, a latent space representation for tasks, such as drawing on a table plane. A control policy learned in this latent space is powerful to regenerate new movements.

The authors are with KTH – Royal Institute of Technology, Stockholm, Sweden, as members of the Computer Vision & Active Perception Lab., Centre for Autonomous Systems, www: http://www.csc.kth.se/cvap, e-mail addresses: {chek,dsong,khubner,danik}@csc.kth.se. This work was supported by EU IST-FP7-IP GRASP, EU IST-FP6-IP-027657 PACO-PLUS, and Swedish Foundation for Strategic Research.

For the robot manipulation task, a recent work [6], adopt a self-supervised, developmental approach where the robot first explores its sensory motor capabilities, and then interacts with objects to learn their affordances. A discrete Bayesian network (BN) [7] is used to capture the statistical dependencies between actions, object features and the observed effects of the actions. Our recent work [8], extended this idea by incorporating task information. The work propose a task constraint model based on a mixed BN, which links the symbolic task requirements to continuous *real* robot sensory data of attributes on both objects and grasping actions. The framework successfully realizes goal-directed imitation in robot grasping tasks. The limitation, however, lies in the inability of the BNs to model high dimensional, multi-channeled sensory data. When the number of nodes which model the sensory streams is high, the training of BNs becomes intractable. Further, the complexity in the structure of the conditional dependencies modeling significantly limits the flexibility of the model.

In current framework, we focus on building a latent task observation model (see the *Latent Variable Model* in Fig. 1). The work relates to that of [5] in that, we want to model a reduced dimensional representation of task. The difference, however, lies in the problem domain: we do not address the movement regeneration as in [5], rather focusing on modeling the task related affordances of objects and robot embodiments. The work can be integrated with the embodiment-specific BNs, as presented in [8] and Fig. 1, to realize goal-directed imitations.

In this paper, we employ the Shared Gaussian Process Latent Variable Model (SGP-LVM) [9], which assumes a factorization of the joint distribution of the data into independent conditional distributions given a shared latent variable. The advantage of such a simple structure is that it allows us to model the conditional dependencies using flexible Gaussian Process mappings (GP) which can be viewed as infinite mixture models whereas in the BN approach we are limited to a fixed set of mixture components. Further, using the SGP-LVM we are not forced to limit the observation space but can leverage the advantage of using the full observed data.

A. Notation

Upper-case letters identify set variables, where Y specifies the full set of all observed variables $Y = \{O, A, C\}$. Superscript ^{S} is used to refer to a sub-set of the variables, $Y^S \subseteq Y$, the letter M_S is the cardinality of the feature subset, $M_S = |Y^S|$. We use letter v to refer to individual variables within a set or a sub-set of the variables. For example $Y_v \in Y$ can be $size$, one of the object features. We use bold letters to

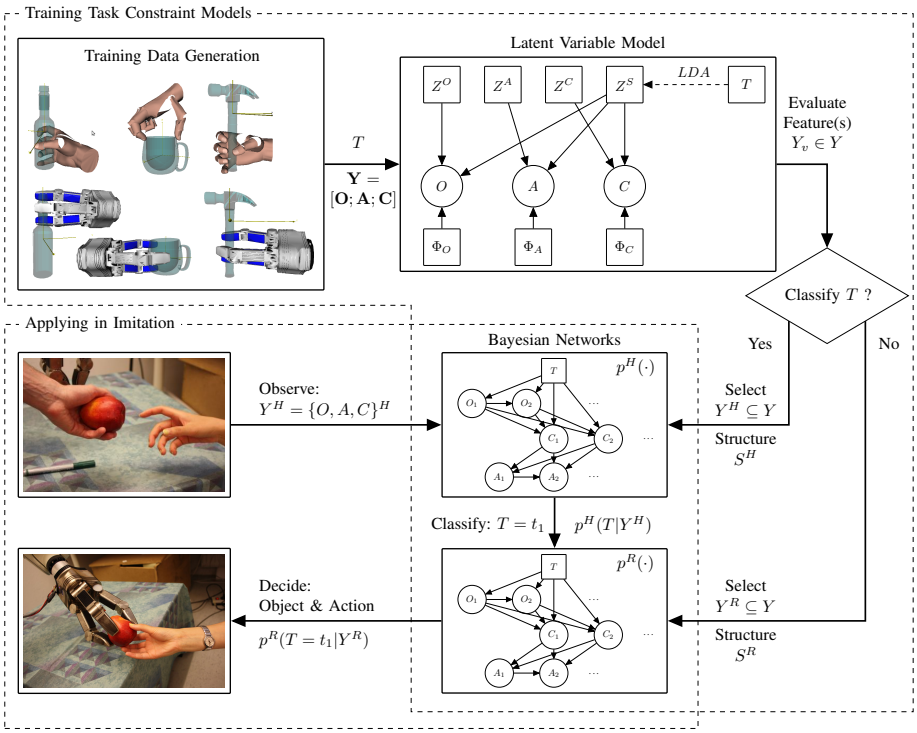


Fig. 1. Schematic System Diagram for the imitation learning framework. The block of *Training Task Constraint Models* shows the learning process of the task constraints from data generated in a simulation environment. This process include *1)* a *Latent Variable Model* that evaluates the underlying structure of the sensory data; and *2)* the embodiment-specific *Bayesian Networks* that model task constraints [8]. The focus of current paper is on the *Latent Variable Model*. It is a Graphical model for learning latent space of three observation streams – $\{O, A, C\}$, capable to reason about the task given any available subset of the input streams. The block of *Applying in Imitation* shows the two-stage imitation using the learned task constraint model. In the first stage the robot observes the scene with the aim of classifying the specific task that the human instructor is performing. Having determined the class posterior given the observed data the robot aims to mimic the action of the instructor.

imply instantiations of a specific variable, where upper-case refers to "all" (or N) instantiations and lower-case refers to a specific instantiation identified by the sub-script index i. For example, $\mathbf{Y} = (\mathbf{y}_1, \ldots, \mathbf{y}_N)^T$ represent N cases of instantiations of all the variables in the set Y. Then, all the N instantiations of variable Y_v will be $\mathbf{Y}^v$, and the i^{th} specific instantiation of Y_v will be $\mathbf{y}_i^v$. The i^{th} specific instantiation of all the variables in Y will be a concatenation of all the variables $\mathbf{y}_i = [\mathbf{y}_i^1; \ldots; \mathbf{y}_i^M]$, where $M = |Y|$. Last, we introduce D to be the dimensionality of an instantiation. Note that $D \geq M$, since M represents the number of possibly multi-dimensional variables.

II. Data Generation

In this subsection, we will briefly introduce how our variables, *i.e.* real feature values, are practically extracted in our system. As it is out of the focus of the paper, we will not describe our grasp planner and the implementation of feature extraction, but refer the reader to [8]. In the notation and Fig. 1 we distinguish between three different types of observed variables defining the training data: *object features* (O) are directly extracted from the object representation, *action features* (A) are directly extracted from the grasp planner, and *constraint features* (C) emerge from the complementation of both, *e.g.* from the resulting contacts. We note that we shortcut the problem of using real world perception for all our features by using a simulation-based architecture and grasp planner. For details on those modules and the tutor-based task labeling process see [8].

For each good grasp i provided by the grasp planner, one training dataset $\mathbf{y_i}$ is generated. While in [8], we used a small network with M=7 features to analyze Bayesian Network learning, we here use M=21 features. This increase of features is not only accompanied by a drastic increase in the dimensionality of the whole feature vector (from D=15 to D=293), but also with strong redundancy in the data. For instance, size (*size*), eccentricity (*ecce*) and shape (*zern*) keep redundant information. These two characteristics (high dimensionality and redundancy) allow us to evaluate our models in terms of dimensionality reduction, dependency detection, and structure learning.

III. Gaussian Process Latent Variable Models

In generative dimensionality reduction the observed data $\mathbf{Y}$ is assumed to have been generated from a low-dimensional latent variable $\mathbf{Z}$ through a mapping f, $\mathbf{y}_i = f(\mathbf{z}_i)$. Assuming the observed data to have been corrupted by additive noise leads to the likelihood of the data,

$$p(\mathbf{Y}) = p(\mathbf{Y}|\mathbf{Z}, f)p(\mathbf{Z})p(f). \quad (1)$$

The Gaussian Process Latent Variable Model (GP-LVM) [10] is a generative model for dimensionality reduction where the generative mapping f is modeled as a product of independent Gaussian Processes (GP) which leads to the

following marginal likelihood,

$$p(\mathbf{Y}|\mathbf{Z},\phi) = \int p(\mathbf{Y}|f)p(f|\mathbf{Z},\phi)df, \tag{2}$$

where ϕ is the hyper-parameters specifying the GP.

In the GP-LVM framework we wish to find the latent locations and the hyper-parameters that maximizes the posterior distribution of the data,

$$p(\mathbf{Z},\phi|\mathbf{Y}) \propto p(\mathbf{Y}|\mathbf{Z},\phi)p(\mathbf{Z})p(\phi). \tag{3}$$

However, as estimating partitioning function is intractable we in practice proceed to minimize the negative log of the product of the margin likelihood the latent prior and the prior over the hyper-parameters. This leads to the following objective,

$$\mathcal{L} = \mathcal{L}_{\text{data}} + \sum_i \ln\, \phi_i + \sum_i \frac{1}{2}||\mathbf{z}_i||^2 + C, \tag{4}$$

where $\mathcal{L}_{\text{data}} = \ln\, p(\mathbf{Y}|\mathbf{Z},\phi)$.

One advantage of the GP-LVM framework is that it is straight-forward to include additional constraints and priors on the latent representation to replace the uninformative prior used in the original formulation Eq. 4.

1) Shared GP-LVM: The Shared GP-LVM (SGP-LVM) [9] is an extension which learns a single latent representation from which two observed data spaces are generated by separate GP's. In this paper, we extend the original SGP-LVM to model the 3 different observation spaces $Y = \{O, A, C\}$. This means that we assume each of the observed data spaces being independent given the latent representation resulting in the following factorization,

$$\begin{aligned} p(\mathbf{Y}) \quad &= \quad p(\mathbf{Y}|\mathbf{Z},\mathbf{\Phi}) = \\ & \quad p(\mathbf{O}|\mathbf{Z},\phi_O)p(\mathbf{A}|\mathbf{Z},\phi_A)p(\mathbf{C}|\mathbf{Z},\phi_C). \end{aligned} \tag{5}$$

Being a generative model with a data-term reflecting reconstruction all the variance in the observed data needs to be represented within the model. The SGP-LVM model models each of the observed data-spaces to have been generated from the same underlying latent variable, this implies an assumption that *all* the variance in each of the observation spaces is shared, variance which is not needs to be "explained away" as Gaussian noise using the noise model. However, for many data-sets it is unlikely for the non-shared variance to be well described using this model leading to a bad fit of the model to the data.

In order to proceed in such situations it was suggested in [9] to extend the SGP-LVM model to also include "private" latent spaces. These spaces models variance in a single observation space which is not shared by the others can be interpreted as "structured noise models". Including such private spaces into our model leads to the following marginal likelihood,

$$\begin{aligned} &p(\mathbf{Y}) = p(\mathbf{Y}|\mathbf{Z}^{\text{S}},\mathbf{Z}^{\text{A}},\mathbf{Z}^{\text{O}},\mathbf{Z}^{\text{C}},\mathbf{\Phi}) = \\ &p(\mathbf{A}|\mathbf{Z}^{\text{S}},\mathbf{Z}^{\text{A}},\phi_A)p(\mathbf{O}|\mathbf{Z}^{\text{S}},\mathbf{Z}^{\text{O}},\phi_O)p(\mathbf{C}|\mathbf{Z}^{\text{S}},\mathbf{Z}^{\text{C}},\phi_C) \end{aligned} \tag{6}$$

In this model the variance which is shared between $\{A, O, C\}$ will be represented using $\mathbf{Z}^{\text{S}}$ while the non-shared variance in each space will be represented by $\mathbf{Z}^{\text{A}}$, $\mathbf{Z}^{\text{O}}$ and $\mathbf{Z}^{\text{C}}$ for A,O and C respectively.

In this project we are not just interested in factorizing the latent representation into variance that is shared from such which is private. Specifically we are interested in variance which is shared *and* that contains task correlated information. Such a factorization can be found by replacing the uninformative prior $p(\mathbf{Z})$ with a distribution that encourages class separation. Such an extension to the standard GP-LVM model was suggested in [11] by including a term based on Linear Discriminant Analysis (LDA).

By noting that our model factorises the latent prior into 4 distinct parts, $p(\mathbf{Z}) = p(\mathbf{Z}^{\text{S}})p(\mathbf{Z}^{\text{A}})p(\mathbf{Z}^{\text{O}})p(\mathbf{Z}^{\text{C}})$. We replace the uninformative prior over the shared space with the LDA prior from [11].

$$p(\mathbf{Z}^{\text{S}}) = \frac{1}{C_S}\text{exp} - \text{tr}\left(\mathbf{S}_w^{-1}\mathbf{S}_b\right), \tag{7}$$

where $\mathbf{S}_w$ and $\mathbf{S}_b$ are the within and between class scatter matrices of the latent representation respectively. In this way we will encourage the shared space to contain information which is shared between the observation spaces and contains class correlated information. Figure shows the graphical model.

2) Training: The objective function above is non-convex and as the solution is sought through gradient based optimization we are dependent on a good initialization of the latent spaces. Further, the dimensionality of the latent spaces are free variables which needs to be estimated from data. These two characteristics of the model poses significant limitations as it for many types of data is non-trivial to estimate the dimensionality and a initialization, specifically for factorized models such as ours.

In recent work [12] a set of regularizers to the SGP-LVM framework has been suggested referred to as FOLS. The FOLS regularizer reduces the demands on the initialization (in practice the latent space is initialized with the observed data) and is capable of learning the dimensionality in addition to factorizing the latent space into non-redundant subspaces. It does so by encouraging solutions with a low-rank covariance matrix and penalize solutions where the shared and the private spaces are non-orthogonal. For a more complete description of the model see [12]. We add the FOLS regularizers to our model which leads to the following objective,

$$\mathcal{L}^{\text{LDA-SGP-LVM-FOLS}} = \mathcal{L}_{\text{SGP-LVM}} + \lambda \cdot \mathcal{L}_{\text{LDA}} + \beta \cdot \mathcal{L}_{\text{FOLS}}. \tag{8}$$

The scalar λ controls the trade-off between reconstruction and class separation and β controls the relative scale of the FOLS regularizer.

The term $\mathcal{L}_{\text{LDA}}$ will encourage a latent representation with large class separation and low within class variance. As noted in [11], the model can be interpreted from two different views. One interpretation is to see $\mathcal{L}_{\text{LDA}}$ as a regularizer on the data-term. However, an equally valid view is to see the data-term as a regularizer on the LDA term. In this paper, we take the later view. By setting λ to a large value we

	A	O	C	A,O	A,C	O,C	A,O,C
NN	78	47	94	88	90	95	92
SVM-Linear	73	81	77	81	82	81	83
SVM-RBF	85	80	93	94	92	94	94
LVM	78	70	89	84	74	92	81

force the shared latent representation to strongly reflect the shared class correlated information in the data. This allows us to evaluate the relative amount of the variance in each observation space which is task correlated.

3) Inference: Training the above presented model means that we have learnt the latent representation $\mathbf{Z} = \{\mathbf{Z}^{\mathrm{S}}, \mathbf{Z}^{\mathrm{A}}, \mathbf{Z}^{\mathrm{O}}, \mathbf{Z}^{\mathrm{C}}\}$ and the hyper-parameters $\mathbf{\Phi} = \{\phi_A, \phi_O, \phi_C\}$ specifying the three generative mappings.

The factorization of joint probability of the observations specified by the model allows us to specify a distribution over any input space given the known latent representation. This means that given any subset of the observation spaces $\mathbf{Y}^S \subseteq \mathbf{Y}$ of the observed features we can infer the latent location by finding the location that maximizes the marginal likelihood,

$$\hat{\mathbf{z}} = \text{argmax} \prod_{v}^{M_S} p(\mathbf{y}^v | \mathbf{z}, \phi_v). \tag{9}$$

The maximum of (9) is found using gradient based methods. This means that the latent locations need to be initialized. In this paper, we initialize the latent location by taking the nearest-neighbors (throughout the experiments in this paper we use 10 neighbors) in the feature training data and initializing using the associated latent location. Our final estimate is the solution corresponding to the highest likelihood solution.

We are interested in inferring the task label associated with a specific subset of the feature observations. To do so, we learn a Gaussian Mixture Model over the shared class-constrained latent space from which the posterior distribution over each class given a latent location can be evaluated. This means that given a location $\mathbf{z}^{\mathrm{S}}$ we can evaluate the conditional distribution for this point to be associated with each task.

IV. Experimental Results

For the experiments with the LDA-SGP-LVM model we selected 900 randomly sampled observation instances $\mathbf{Y}$ uniformly distributed over task. We used 450 instances for training and the remaining to test the model.

A. Task Classification

We first test our models performance in classifying the observed task, we use the MAP solution under the GMM model over the latent space. To understand the effect of the classification performance with respect to different observation domains we perform the classification on each permutation of the observed features, the results are compared to a set of baseline algorithms in Tab. IV-A.

Compared to the purely discriminative base-line algorithms we expect our model to perform worse as it is both learning a representation for task discrimination and for reconstruction and generation. Further, in our model it is possible to infer the task from any observed subspace while for the base-line algorithms a separate model needs to trained for each input combination.

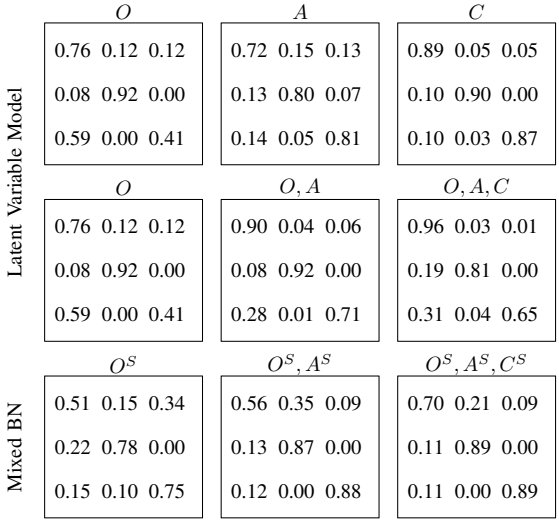


Fig. 2. Confusion matrices on classification of 3 tasks, *hand-over, pouring, tool-use*, given observation on different sub-sets features in $\{O, A, C\}$. The first two row are the results from the *Latent Variable Model*. The third row is the results from the *Mixed BN* in previous work [8]. The superscript ^{S} represent subset of the features, *e.g.* $O^S \subset O$

Taking this into consideration we believe that our model performs competitively.

In Fig. 2 the confusion matrices for task classification for the proposed model and our previous BN based model is shown. The first row shows the results using each of the observed features in turn. We can see that the object features are generally good at discriminating the *hand-over* and *pouring* task but confuses *tool-use* with *hand-over*. The action features in comparison provides a good indications for each task but are worse than the object features in discriminating pouring. This is to be expected as the objects that are pour-able are quite specific while the pouring action itself can be ambiguous. Finally, for the constraint features we see that compared to object the are capable to clearly discriminate the task of *tool-use*. This is also something we would expect as this task poses quite specific constraints.

Comparing the result of the proposed model with our previous work we can see that when we only have a limited available our model performs significantly better while when given additional input features our models performance drops. This is an indication of the mismatch between the structure of the dependencies in our model compared to the data, where the single layer model is too simplistic while the more complex BN model is closer to the actual data.

B. Different Features Show Different Properties

By incorporating the FOLS regularizers into the learning framework we are able to infer the dimensionality of the latent factorization from data. In our experiments this resulted in a model with 2 shared dimensions and 5,2 and 2 private dimensions to represent A,O and C respectively.

In Fig. 3 we see the *action* features containing the largest proportion of non-task correlated variance among our three

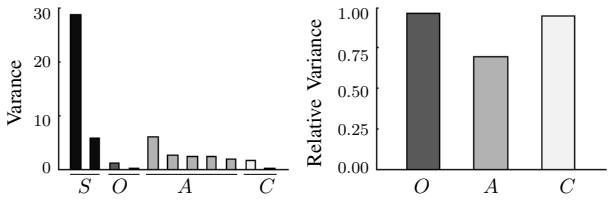


Fig. 3. In the left-most plot shows the variance along each dimension of the latent representation. The right plot shows the proportion of the variance in each of the separate latent spaces which is contained in the shared space.

feature classes. This is not surprising as we expect there to be a larger variability in the manner an action is performed compared to the object and constraints for a specific task.

For imitation learning we are interested in learning what features are relevant and what combination of features reduces the entropy of the task distribution the most. However, this is clearly task dependent as say for example knowing the object is not likely to be particularly discriminative between the task of pouring and hand-over while providing very useful knowledge for choosing between pouring and tool-use. If such information could be extracted we would be able to direct the robots perception in such manner to acquire information to maximally reduce the entropy over tasks. Our model being generative we can exploit the *shared-private* latent structure to extract such information.

For an observed data point $\mathbf{y}_i^v$, we can evaluate the marginal likelihood $p(\mathbf{y}_i^v|\mathbf{z}_i, \mathbf{Z})$ of the latent location under the model. In the model this factorizes as follows,

$$p(\mathbf{y}_i^v|\mathbf{z}_i, \mathbf{Z}) = p(\mathbf{y}_i^v|\mathbf{z}_i^{\mathrm{S}}, \mathbf{z}^{\mathrm{v}}, \mathbf{Z}^{\mathrm{S}}, \mathbf{Z}^{\mathrm{v}}). \tag{10}$$

As the location in the shared latent sub-space $\mathbf{z}_i^{\mathrm{S}}$ represents task correlated variance in the data we can by fixing the location in the private subspace $\mathbf{z}_i^{\mathrm{v}}$ (which represents the non-shared non-task correlated variance) evaluate the effect of the marginal likelihood with respect to the shared variable for different observation features.

In Fig. 4 the result of uniformly sampling the marginal likelihood over the shared latent space while keeping the private location fixed for the three different tasks and observation features are shown. The first row shows how the likelihood varies for an instance of the *hand-over* task. In the first column the likelihood for the objective features are shown, it can be seen that the probability mass is distributed over all training data locations. This is to be expected as the task *hand-over* can be performed on every object. The middle column shows the results for the action features, which have a large region associated with a high and two regions with low likelihood values. Identifying the training data location we can see that the region of high-likelihood corresponds to hand-over while the low-likelihood regions correspond to pouring. This is sensible as the action performed when pouring is significantly different to the one for hand-over. In the last column the distribution for the constraint model is shown. As we can see the distribution have a very high entropy being nearly completely flat which implies that there is very little information in the constraints for the task of hand-over.

By similar reasoning, we can see in the second row that for the task of *pouring* the object features are less discriminative while the action features places a high-probability mass around an area occupied by pouring and tool-use instances. More importantly, here, the constraint features are very discriminative with very dominant mode which identifies the task as pouring.

Finally, in the third row, the results for *tool-use* is shown. Here we can see that each of the three features are multi-modal and all provide relevant information to reason about the task.

Further, for each of the three examples shown above we note that the distribution over the action features is a lot less peaked in comparison with the object and action features. This is to be expected as there is a much larger variability in realizing a grasp compared to object and constraint features.

The above results clearly exemplifies the strengths of our model, by learning a factorized model we can evaluate given an observation in one feature space which feature we should acquire in order to reduce the entropy in the decision of task the most. Further, observing how the probability mass is distributed over the shared latent space gives a notion of the generative qualities of our model. The object and constraint distributions are tightly centered around the data while for the action features it is more broadly distributed. This is good as the first two often poses next to *hard* constraints on the grasp while for the action this shows that there are several possible ways to realize a specific action.

V. Conclusion

In this paper we have presented an extension to the Shared Gaussian Process Latent Variable model to more than two observation spaces and by incorporating partial discriminative constraints. Further, we have including recently proposed latent regularizers in order to learn a non-redundant latent representation of the data.

The proposed model is applied to factorize the joint distribution of a set of different features for the robotic task of grasping. Our generative model performs comparatively to purely discriminative methods for task classification. Further, by exploiting the generative nature of our model we are able to evaluate the discriminative power of different features to different tasks, something which can be very useful in scenarios with limited observations.

Robots and humans have significantly different embodiments, this means that the features that are important for humans and robots in order to perform a specific task successfully are likely to be different. In our previous work we bridge this gap by using separate models for the robot and the human, where classification is done in the human model and generation in the robotic model. However, given corresponding instances of robot and human action, the framework developed above is capable of simultaneously learning a latent representation linking the two different embodiments making it possible to contain the full framework whit-in a single model. This is something we would to evaluate in future work.

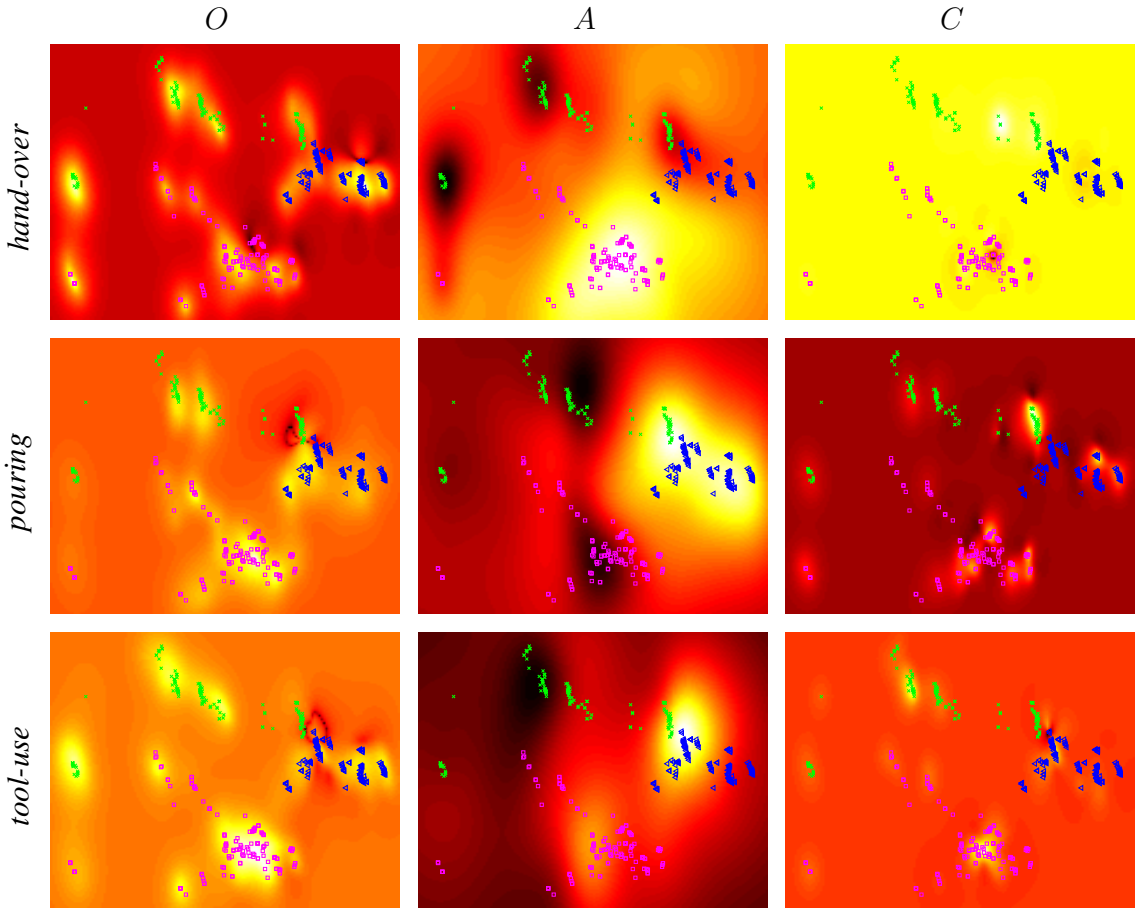


Fig. 4. Samples of the likelihood under the model. Each images is generated by fixing the location in the private-non-task correlated latent subspace and uniformly sampling over the shared-task correlated. Each row represents a specific task, top to bottom: *Hand-Over*, *Pouring* and *Tool-Use*. The columns represents the three different observation spaces, left to right: *Object*,*Action* and *Constraint*. The location of the training data is shown over the plots, where *Magenta*,*Green* and *Blue* indicates *hand-over, pouring* and *tool-use* respectively. Note due to the intractability in estimating the partioning functions the actual value of the posterior should not be interpreted only the within observation space scale is relevant. For visualization purposes we have scaled and normalized the plots for ease of interpretation.

In this paper we have for clarity of presentation focused on the discriminative power of the model, however, the model is generative and capable of reconstruction outside the training data. As noted in the expremental section, there is indications of the good generative qualities. This is something we will evaluate in upcoming work.

REFERENCES

[1] C. L. Nehaniv and K. Dautenhahn, Eds., *Imitation and Social Learning in Robots, Humans, and Animals: Behavioural, Social and Communicative Dimensions*. Cambridge University Press, 2004.

[2] J. Bohg, C. Barck-Holst, K. Huebner, M. Ralph, B. Rasolzadeh, D. Song, and D. Kragic, *Towards Grasp-Oriented Visual Perception for Humanoid Robots*. Computational Vision and Active Perception, School of Computer Science and Communication, Royal Institute of Technology-KTH, 2009.

[3] J. Romero, H. Kjellström, and D. Kragic, "Modeling and evaluation of human-to-robot mapping of grasps," *International Conference on Advanced Robotics (ICAR)*, 2009.

[4] M. Toussaint, N. Plath, T. Lang, and N. Jetchev, "Integrated Motor Control, Planning, Grasping and High-level Reasoning in a Blocks World using Probabilistic Inference," in *ICRA*, 2010.

[5] S. Bitzer, I. Havoutis, and S. Vijayakumar, "Synthesising Novel Movements Through Latent Space Modulation of Scalable Control Policies," in *SAB*, 2008.

[6] L. Montesano, M. Lopes, A. Bernardino, and J. Santos-Victor, "Learning Object Affordances: From Sensory–Motor Coordination to Imitation," *IEEE Transactions on Robotics*, pp. 15–26, February 2008.

[7] J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, September 1988.

[8] D. Song, K. Huebner, V. Kyrki, and D. Kragic, "Learning Task Constraints for Robot Grasping using Graphical Models," in *IEEE International Conference on Intelligent Robots and Systems 2010*, 2010.

[9] C. H. Ek, "Shared Gaussian Process Latent Variable Models," Ph.D. dissertation, Oxford Brookes University, 2009.

[10] N. D. Lawrence, "Probabilistic non-linear principal component analysis with Gaussian process latent variable models," *The Journal of Machine Learning Research*, vol. 6, pp. 1783–1816, November 2005.

[11] R. Urtasun and T. Darrell, "Discriminative Gaussian process latent variable model for classification," *Proceedings of the 24th international conference on Machine learning*, pp. 927–934, 2007.

[12] M. Salzmann, C. H. Ek, R. Urtasun, and T. Darrell, "Factorized orthogonal latent spaces," in *Proceedings of the Thirtenth International Conference on Artificial Intelligence and Statistics*, May 2010.

Multivariate Discretization for Bayesian Network Structure Learning in Robot Grasping

Dan Song, Carl Henrik Ek, Kai Huebner and Danica Kragic

***Abstract*— A major challenge in modeling with BNs is learning the structure from both discrete and multivariate continuous data. A common approach in such situations is to discretize continuous data before structure learning. However efficient methods to discretize high-dimensional variables are largely lacking. This paper presents a novel method specifically aiming at discretization of high-dimensional, high-correlated data. The method consists of two integrated steps: non-linear dimensionality reduction using sparse Gaussian process latent variable models, and discretization by application of a mixture model. The model is fully probabilistic and capable to facilitate structure learning from discretized data, while at the same time retain the continuous representation. We evaluate the effectiveness of the method in the domain of robot grasping. Compared with traditional discretization schemes, our model excels both in task classification and prediction of hand grasp configurations. Further, being a fully probabilistic model it handles uncertainty in the data and can easily be integrated into other frameworks in a principled manner.**

I. Introduction

In the research field of robot grasping and manipulation, an important challenge is to automatically plan a grasp on an object that affords an assigned manipulation task. This requires a robot to integrate a large range of sensory streams in order to estimate the state of the scene, understand the task requirements, and reason about its ability to plan and control in an uncertain, open-ended environment. In recent years, learning by imitation has been one important approach to these problems [1], [2], [3], [4]. The goal is to design a system whereby a robot learns a task by imitating a human teacher. In order to observe a human demonstration, the robot needs to obtain the state of the entire scene, not only the objects, but also the human actions which often are characterized using many variables (e.g. hand kinematics). In addition, the robot also needs to recognize the intention of the human to be able to reproduce the grasp in a goal-directed way. This poses a significant modelling challenge.

To correctly describe a manipulation task, both conceptual high-level information and continuous low-level sensorimotor variables are needed. In previous work [5], we have applied Bayesian Networks (BN) [6] to model the grasping scenario. The aim is to exploit the strengh of BNs in modeling complex joint distribution of these variables, so that the robot is able to reason at both high-level task representations and low-level planning and control using its sensorimotor systems. Though the results are promising, the approach suffers from inheritant disadvantages of BN, being able to use only a small sub-set of the available sensory streams.

These difficulties arise from several sources. When many nodes are included, manually encoding the structure of the network is non-trivial and poses significant challenges. Therefore, we are motivated to learn the BN structure automatically from the data. A large body of work exists (see review in [7]) learns the structure of BN in scenarios where all the variables are discrete. However, our observations are represented using both discrete and multivariate continuous variables. In such scenarios, no directly applicable method for learning the structure exists. To learn structure from continuous data, several approaches have sought to discretize continuous data as a prestep (see reviews in [8]). However, as data-dimensions increases discretisation becomes significantly more challenging. Recently, a multivariate discretization scheme was developed in [9]. However, the method suffers from ineffective clustering in high-dimensional spaces and slow convergence.

The contribution in this paper is to introduce a novel multi-variate discretization approach, which makes use of high correlation of the high dimensional data, combining a non-parametric dimensionality reduction approach (Gaussian process latent variable model (GP-LVM) [10]), with a Gaussian mixture model. In addition, the GP-LVM being a generative model, when combined with BNs, can improve the data reconstruction in the high-dimensional *continuous* space. We evaluate our proposed discretization approach by examining its effect on learning BNs. When compared with other alternative approaches, the GP-LVM-based method significantly improves the task classification accuracy, and intuitive data reconstruction in high-dimensional space.

II. Related Work

Graphical models such as Bayesian networks [6] aim at exploiting conditional independencies in the data in order to factorize the joint distribution of the data. For interpretable scenarios such as object manipulation, the factorization encodes the semantic relationships between groups of variables that represents the entire scene of manipulation tasks. Further, being a generative model, a BN can be directly applied for reasoning and planning of new grasps outside the training data. In [3] a discrete BN is applied to model affordances of objects while a robot is interacting with them in simple manipulation tasks to explore its sensorimotor capabilities. In our recent work [5], we extended the domain to complex,

D. Song, C.H Ek, K. Huebner and D. Kragic are with KTH – Royal Institute of Technology, Stockholm, Sweden, as members of the Computer Vision & Active Perception Lab., Centre for Autonomous Systems, www: http://www.csc.kth.se/cvap, e-mail addresses: {dsong,chek,khubner,danik}@csc.kth.se.

task-oriented grasp planning problems, and tackled challenges in both object/task recognitions and grasp selection under uncertain environments. The model allows not only to reason about high-level task representations, but also to make detailed decisions on which object to use and which grasp to apply in order to fulfill the requirements of an assigned task. However, the BN models both discrete and continuous variables, which presents the limitations when a large number of variables are to be modeled, and when the structure of the network is to be learned.

Learning BN structure from both continuous and discrete data is difficult, particularly when continuous data is high-dimensional and has complex distributions. Most algorithms for structure learning only work with discrete variables. Therefore, a common approach is to convert the mixed modeling scenario into a completely discrete one by discretizing the continuous variables [8]. In data discretization, most techniques are based on either heuristic evaluation [3], or equal-frequency and equal-width binning methods [7], [8], [11]. In latter, the data is divided into a set of bins, and then the number and the location of the bins are optimized based on some information criterion such as Akaike's criterion [7], [12]. However, the method is only applicable for continuous variables with only one dimension. Multivariate discretization has been explored for association rule discovery in the field of data mining. A recent work in [9] introduced a novel approach that is based on density-based clustering techniques. The method assumes that high density data clusters often occur when there are high associations between the continuous variables. Once these clusters were identified, a genetic algorithm was applied to optimize the cut points for the discretization intervals.

The idea behind the method in [9] is similar to ours in that we also use the clusters discovered in the data as the class information used for discretization. However, [9] explored the clustering in the original observation space of the data which will suffer when the variable is very high-dimensional and thus density learning becomes inefficient. In robotic applications, many variables of interest such as hand joint kinematics are high-dimensional. However, the intrinsic dimensionality or the number of degrees of freedom are often much smaller due to significant between-dimension correlation in the observed space. In dimensionality reduction the aim is to exploit such correlation in order to find a more compact and efficient representation.

The field of dimensionality reduction has received significant attention over the last decade. Originating from simple linear algorithms such as Multi-Dimensional-Scaling [13] and Principle Component Analysis, several non-linear extensions were suggested [14], [15]. However, these methods often rely on assumptions that significantly limits their application. Therefore more general, generative interpretations such as Probabilistic PCA (PPCA) [16] were proposed. Of particular success has been the GP-LVM being a non-linear generalization of PPCA based on non-parametric regression using Gaussian Processes (GP). In next two sections, we will first introduce GPs and the GP-LVM. Based on this background, we will proceed to explain our proposed discretization model.

III. Gaussian Processes

A GP [17] is defined as a collection of random variables, any finite number of which follows a joint Gaussian distribution. GPs are commonly used to model distributions over functional spaces making probabilistic treatment of such possible.

Given a set of data $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_N]^{\mathrm{T}}$, $\mathbf{x}_i \in \Re^q$ and $\mathbf{Y} = [y_1, \ldots, y_N]^{\mathrm{T}}$, $y_i \in \Re$, we assume that y_i are related to $\mathbf{x}_i$ through an underlying functional relationship f where the observations $\mathbf{Y}$ have been corrupted by additive Gaussian noise,

$$y_i = f(\mathbf{x}_i) + \epsilon, \tag{1}$$

where $\epsilon \sim \mathcal{N}(0, \sigma^2)$. We assume that the unknown function values f can be modeled using a GP,

$$p(f(\mathbf{X})|\mathbf{X}) = \mathcal{N}\big(\mu(\mathbf{X}), k(\mathbf{X}, \mathbf{X})\big), \tag{2}$$

where $\mu(\cdot)$ is the mean and $k(\cdot, \cdot)$ the covariance function respectively. Without loss of generality we can remove the bias from the data and set the mean funtion to be constant zero. The covariance function defines what types of functions are more prominent in the prior, and are specified using a set of parameters $\mathbf{\Phi}$ which will be referred to as the hyper-parameters of the GP. The hyper-parameters are difficult to know a-priori, so we want to learn them from data. Combining the prior with the likelihood, which is Gaussian due to the noise assumption, and marginalizing,

$$p(\mathbf{Y}|\mathbf{X}, \mathbf{\Phi}) = \int p(\mathbf{Y}|f(\mathbf{X}))p(f(\mathbf{X})|\mathbf{X}, \mathbf{\Phi})df(\mathbf{X}), \tag{3}$$

leads to the marginal likelihood of the observed data. We can then find the hyper-parameter $\hat{\mathbf{\Phi}}$ that maximizes Eq.3.

Through the definition of a GP, the joint distribution of $\mathbf{X}$ and a set of new input locations $\mathbf{X}^*$ can be formulated. Marginalization of the observed data $\mathbf{X}$ leads to the predictive distribution,

$$p(f(\mathbf{X}^*)|\mathbf{Y}, \mathbf{X}, \mathbf{\Phi}) = \mathcal{N}\big(k(\mathbf{X}^*, \mathbf{X})\left(k(\mathbf{X}, \mathbf{X}) + \sigma^2\mathbf{I}\right)^{-1}\mathbf{Y}, \\ k(\mathbf{X}^*, \mathbf{X}^*) - k(\mathbf{X}^*, \mathbf{X})\left(k(\mathbf{X}, \mathbf{X}) + \sigma^2\mathbf{I}\right)^{-1}k(\mathbf{X}, \mathbf{X}^*)\big), \tag{4}$$

from which inference can be performed.

As can be seen above, learning in the GP framework requires invertion of a $\mathrm{N} \times \mathrm{N}$ matrix, an operation of cubic complexity. This places significant constraints on the size of the datasets for which the framework can be applied. This has led to a significant amount of work on methods which aim to reduce the computational cost associated with the model. In specific, a set of methods which introduce an additional set of input variables $\mathbf{U}$ referred to as the *inducing* variables have been proposed. By assuming the observed data to be independent given the inducing points, the prior can be written as $p(f(\mathbf{X})) = \int p(f(\mathbf{X})|\mathbf{U})p(\mathbf{U})d\mathbf{U}$. When substituted into the derivation of the predictive distribution, the inducing conditional takes the following form,

$$p(f(\mathbf{X})|\mathbf{U}) = \mathcal{N}\big(k(\mathbf{X}, \mathbf{U})k(\mathbf{U}, \mathbf{U})^{-1}\mathbf{U}, \\ k(\mathbf{X}, \mathbf{X}) - k(\mathbf{X}, \mathbf{U})k(\mathbf{U}, \mathbf{U})^{-1}k(\mathbf{U}, \mathbf{X})\big). \tag{5}$$

Note that the matrix inversion in Eq. 5 is only applied to the covariance matrix on the inducing points.

By using a much smaller number of inducing points compared to the original data, several different approximations have been suggested. Explaining the different approximation in detail is beyond the scope of this paper. However, we briefly note that one kind of approximation method adds the inducing points as parameters of the covariance function and estimates their location jointly with the remaining hyper-parameters. Intuitive as this might be, it significantly increases the number of parameters to be estimated which makes such an approach prone to over-fitting. A recent work by [18] suggested a variational framework to circumvent this problem. The method treats the inducing points as parameters, and proceeds by minimizing the Kullback-Leibler divergence between the exact posterior distribution and the variational distribution. In this paper we will make use of this variational approximation in [18].

The GP-LVM [10] is a generative model for dimensionality reduction based on GPs. We assume a set of observed data $\mathbf{Y} = [\mathbf{y}_1, \ldots, \mathbf{y}_N]^{\mathrm{T}}$, with $\mathbf{y}_i \in \Re^D$, is generated from a low-dimensional variable $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_N]$, with $\mathbf{x}_i \in \Re^q$, through a mapping f corrupted by additive Gaussian noise, $\mathbf{y}_i = f(\mathbf{x}_i) + \epsilon$, where $\epsilon \sim \mathcal{N}(\mathbf{0}, \sigma^2\mathbf{I})$. Each dimension of the output is modeled as an independent GP,

$$p(f(\mathbf{X})|\mathbf{X}) = \prod_i^D p(f(\mathbf{X})^{\mathrm{T}}\mathbf{e}_i|\mathbf{X})) \tag{6}$$

where $\mathbf{e}_i$ is standard basis vector. The GP-LVM proceeds in analog with the regression framework with one significant difference. In the GP-LVM the input variables $\mathbf{X}$ are treated as random variables, and the solution to $\mathbf{X}$ is found together with the hyper-parameters through maximum likelihood,

$$\{\hat{\mathbf{\Phi}}, \hat{\mathbf{X}}\} = \mathrm{argmax}_{\{\mathbf{\Phi},\mathbf{X}\}} p(\mathbf{Y}|\mathbf{X}, \mathbf{\Phi}). \tag{7}$$

This might seem non-sensical as clearly the combination of mappings f and input locations $\mathbf{X}$ that could have generated the observed data $\mathbf{Y}$ is infinite. However, by fixing the dimensionality of the latent space q to be significantly smaller compared to the observed data, and by the regularizing effect of the GP-prior, a solution can be found efficiently.

IV. GP-LVM Discretization

In this section, we explain a novel descretization approach, which, by exploiting recent advances in sparse GPs and the GP-LVM, is able to compactly represent high-dimensional continuous data using a low-dimensional discrete mixture model in a principled way.

Our approach is a straight-forward two-stage framework, as exemplified in Fig. 1. Given a set of observed high-dimensional data $\mathbf{Y}$, in this example the 20D feature $fcon$, we wish to find a compact discrete representation of the data which can be efficiently used in a BN. In the first stage we use a GP-LVM model where the generative mapping is formulated as a sparse GP from a small sub-set of inducing points. Using the sparse variational approximation of [18], the location of the inducing points (the red star markers in Fig. 1 a) and the hyper-parameters of the generative mapping can be found. Further, in order to avoid setting the dimensionality by hand and reduce reliance on initialization, we include the rank-regularization framework of [19].

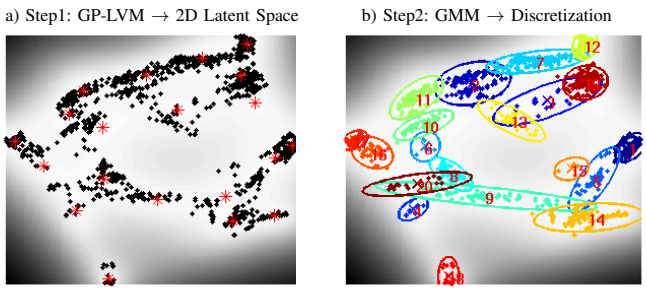


Fig. 1. GP-LVM-based discretization for $fcon$: a) 2D latent space learned using sparce GP-LVM, b) discretization using GMM in the 2D latent space.

In the second stage, one could directly use the inducing points of the GP as cluster-centers to discretize the data. However, when training GP-LVM, we use a spherical covariance function in order to reduce the number of parameters that need to be estimated from data. But this might lead to a less compact representation since variances in the dimensions with high correlation will be modeled as the same as those with low correlations. Our solution is, in the second stage, to learn a more compact representation using a mixture model that allows full covarances therefore more flexible component representations. More specifically, we learn a Gaussian mixture model (GMM) explaining the latent locations $\mathbf{X}$. The inducing points are used as the initial centers of all mixture components, and the model is optimized through the standard Expectaion Maximization approach. Fig. 1 b) shows the resulted mixture model, with the ellipsoids representing one standard deviation of the Gaussian components, and the colored data points being the resulted discretization.

For runtime evaluation, an unoptimized Matlab implementation with 1800 data points, it takes about 100 seconds to train the discretization model for the worst case data.

V. Data Generation

In this section, we will briefly introduce how the variables, i.e. the feature values, are extracted. Tab. I shows the features used in this work. The features describing each grasp are divided into three sub-sets: *object features* (O) extracted from the object representation, *action features* (A) extracted from the planned grasps, and *constraint features* (C) resulting from the complementation of both, i.e. the hand-object configuration. Each grasp is visualized in GraspIt! to a human tutor who associates it with a task label (T).

Fig. 2 shows the schematic of the data generation process. To extract those features, we first generate grasp hypotheses using the grasp-planner BADGr [20], and evaluate them as scenes of object-grasp configurations in a grasp simulator, GraspIt! [21]. BADGr includes extraction and labeling modules to provide the set of variables presented in Tab. I. The interested reader is referred to [5], [20] for more details on the feature extraction. We emphasize that the grasp representation does not have to be non-redundant, e.g. *cvex* and *shcv* are allowed variables to both represent object shapes. Such an "over-representation" of the featured

TABLE I

USED FEATURES WITH THEIR DIMENSIONALITY D (FOR CONTINUOUS) AND NUMBER OF STATES N AFTER DISCRETIZATION.

	Name	D	N	Description
T	$task$	-	3	Task Identifier
O_1	$obcl$	-	6	Object Class
O_2	$size$	3	8	Object Dimensions
O_3	$cvex$	1	4	Convexity Value $[0, 1]$
O_4	$shcv$	3	7	Shape Class Vector (Zernike Similarity)
A_1	$fcon$	20	20	Final Hand Configuration
A_2	dir	4	20	Approach Direction (Quaternion)
A_3	pos	3	14	Grasp Position
A_4	$egpc$	2	6	Eigengrasp Pre-Configuration
A_5	$upos$	3	11	Unified Spherical Grasp Position
C_1	$fvol$	1	6	Free Volume
C_2	$gbvl$	1	4	Volume of Grasped Boxes
C_3	$pshcv$	3	7	Part Shape Class Vector (Zernike Similarity)
C_4	$pecce$	1	3	Part Eccentricity $[0, 1]$
C_5	$g1bx$	1	2	Grasped-1-Box Value $[0, 1]$
C_6	$qeps$	1	5	Grasp Stability Measure (eps)
C_7	$qvol$	1	3	Grasp Stability Measure (vol)

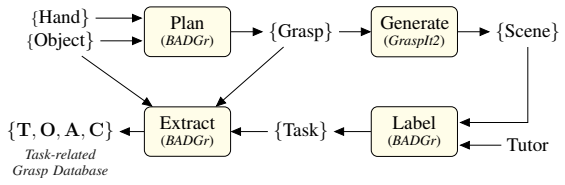


Fig. 2. Schematic diagram for generating task-related grasp database.

variables allows us to use BNs to identify the importance of, and dependencies between these variables in the scenarios of robot grasping and manipulation.

VI. RESULTS

In this section, we will describe four experiments to evaluate our framework. In order to compare the suggested approach with other methods we also present results obtained by discretizing using a mixture model in either the observed data space (NoReduce), or in the space spanned by the dominant principle components (PCA). To compare them under the same level of model complexity, the dimensionality of the mixture model and the number of principle components are set to be the same as for those learned by the GP-LVM.

To generate data, the grasp hypotheses are produced on 24 object models evenly covering six object classes ($obcl$) including knifes, hammers, screwdrivers, glasses, bottles and mugs. The grasps are labeled according to three manipulation tasks: *hand-over, pouring* and *tool-use*. The total data-set consists of 1800 data points uniformly divided over the three tasks. We use 80% of the data for training and the remainng 360 instances for testing.

In all the experiments, when feature set O is observed, all object features except the object class $obcl$ are observed. We assume that $obcl$ is unknown in order to simulate the real-world scenarios where recognizing object categories from its raw features is still a hard problem for robot sensor systems.

A. Experiment I: Structure Learning

The first experiment is to learn structure of BNs from the discretized data by using the three different discretization schemes. We use a greedy search algorithm to find the structure, or the directed acyclic graph (DAG), in a neigborhood of graphs that maximizes the network score (Bayesian information criterion [22]). The search is local and in the space of DAGs, so the effectiveness of the algorithm relies on the initial DAG. As suggested by [7], we use another simpler algorithm, the maximum weight spanning tree [23], to find an oriented tree structure as the initial DAG. We assume the task class variable is the 'cause' of the systems thus the root node of the network.

Fig. 3 shows the results of learned DAGs from the three models. We note that learning the structure from data reveals complicated relationships among these large pool of variables, which will otherwise be very difficult to encode by human experts. From an initial inspection, the DAGs associated with the different discretization schemes share much of their structure. And they conform to our intuitive knowledge of the dependency relations between the variables. For example, the three action features – pos, $upos$ and dir – are fully connected because the unified spherical grasp position $upos$ is directly derived from the grasp position pos and the hand orientation with respect to the object dir. We also see that $task$ has a direct connection to $obcl$ which in turn directly determines the other object features. This make perfect sense as among the six object classes, 3 classes are tools therefore afford the *tool-use* task, and the other 3 classes are containers therefore afford the *pouring* task.

When comparing the DAG of GP-LVM to those of PCA and NoReduce models, the main difference lies in the connections involving $fcon$. This is not surprising. As $fcon$ represents the final grasp configuration of the hand which is high-dimensional and contains a lot of information, it is therefore expected to be most affected by the discretization methods. As we can see, $fcon$ in the GP-LVM model has more parents thus more family members than that of the other two models. This indicates that the GP-LVM-based discretization of $fcon$ captured a different representation in the continuous, high-dimensional space of the variable, therefore, the dependencies of $fcon$ with other variables are different. In specific, in the GP-LVM model, $fcon$ is connected to $pecce$, the eccentricity of the grasped part, which clearly has a big impact on the final grasp configuration. This dependency is discovered when using GP-LVM-based discretization scheme, but is lost by both PCA and NoReduce models.

B. Experiment II: Task Classification

As we do not have the true structure of the BN to evaluate the learned DAGs, in this section we evaluate the effectiveness of the structure learning using the three different discretization schemes by comparing their task classification performance. As shown in Fig. 4, this task classification is based on the inference of $task$ variable given observation of different set of other variables that form a complete permutation of the 3 feature sub-sets: O, A and C.

There are three major observations from the result of this

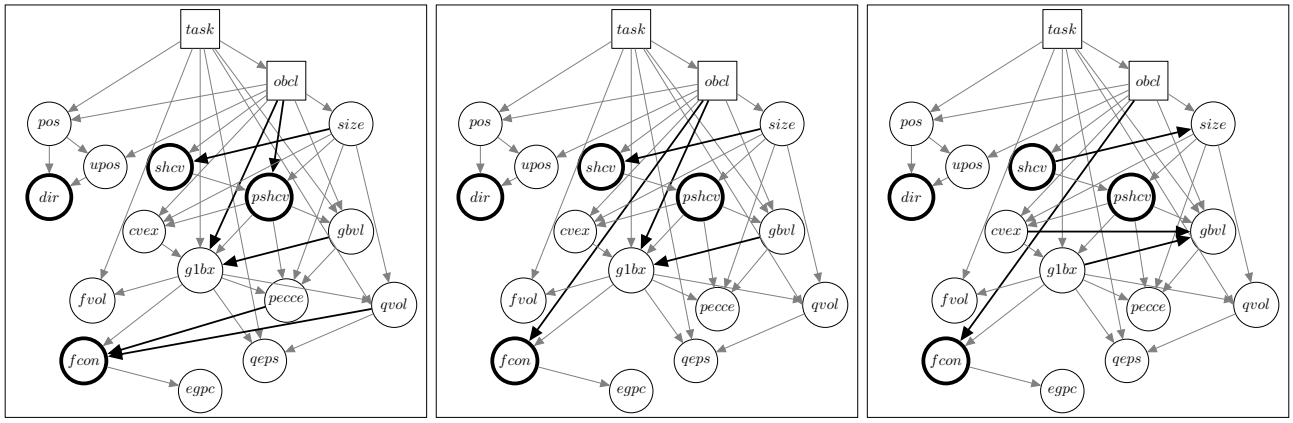


Fig. 3. Experiment I: The three resulting DAGs by applying structural learning on (left) GP-LVM, (middle) PCA and (right) NoReduce discretization schemes. The differences in DAGs are highlighted by thick arrows. Square nodes represent discrete variables and circled nodes continuous. Continuous nodes with thick boundaries identify high-dimensional variables onto which different discretization schemes have been applied.

	O	A	C	O, A	O, C	A, C	O, A, C
GP-LVM	0.70 0.12 0.17 0.12 0.88 0.00 0.16 0.00 0.84	0.58 0.10 0.32 0.03 0.38 0.59 0.01 0.05 0.94	0.75 0.10 0.15 0.33 0.57 0.10 0.38 0.00 0.62	0.74 0.09 0.17 0.05 0.95 0.00 0.03 0.00 0.97	0.73 0.04 0.23 0.29 0.71 0.00 0.11 0.00 0.89	0.78 0.12 0.10 0.16 0.82 0.02 0.03 0.03 0.94	0.86 0.04 0.10 0.15 0.85 0.00 0.04 0.01 0.95
PCA	0.71 0.13 0.16 0.11 0.89 0.00 0.40 0.00 0.60	0.58 0.13 0.29 0.02 0.59 0.39 0.02 0.10 0.88	0.73 0.06 0.21 0.47 0.46 0.07 0.62 0.00 0.38	0.77 0.12 0.11 0.03 0.97 0.00 0.14 0.00 0.86	0.78 0.04 0.18 0.41 0.57 0.02 0.36 0.00 0.64	0.79 0.08 0.13 0.32 0.65 0.03 0.18 0.00 0.82	0.87 0.06 0.07 0.26 0.73 0.01 0.13 0.00 0.87
NoReduce	0.58 0.28 0.14 0.28 0.44 0.28 0.09 0.19 0.72	0.69 0.19 0.12 0.12 0.53 0.35 0.17 0.36 0.47	0.58 0.07 0.35 0.14 0.17 0.69 0.46 0.01 0.53	0.68 0.21 0.11 0.11 0.50 0.39 0.06 0.20 0.74	0.72 0.11 0.17 0.55 0.22 0.23 0.37 0.02 0.61	0.77 0.12 0.11 0.16 0.44 0.40 0.45 0.17 0.38	0.79 0.11 0.10 0.32 0.51 0.17 0.29 0.14 0.57

Fig. 4. Experiment II: Confusion matrices for task classification given different observations spaces: permutations of O, A, C features. For each 3×3 matrix, from left to right (top to down), the 3 tasks are: *hand-over, pouring, tool-use*.

experiment: *i)* Object features contain a lot information that is task-relavant. By just observing O using GP-LVM model, we see quite good classification rate. But A and C features by themselves are close to random. When more information is observed, the classification rate improves. Particularly, for GP-LVM and PCA models, O and A compliments each other and improve the classification rate significantly for the *pouring* and *tool-use* tasks. When C is also added in the observation space (O, A, C), the classification for the *hand-over* task is improved in both models. However, we notice that the accuracy on classifying *pouring* task decreases from 95% to 85%. This may be explained by the imbalanced training data in many of the C variables. For example, when there is lack of data points for some states of the C variables when the task is *pouring*, the conditional probabilities learned in the BNs will be under-determined. In such situations, clamping (observing) these C variables at these states will certainly hurt the inference on the posterior distribution on task. *ii)* When comparing the three models, NoReduce, which discretizes the variables in original high-dimensional space, has close-to-random classification rate. The two models with dimensionality reduction (GP-LVM, PCA) greatly outperform the NoReduce model. GP-LVM is clearly the best in most of the observation conditions. *iii)* When only A features are observed, we see quite high classification rate only for *tool-use* (PCA: 88%, GP-LVM: 94%), while the other two tasks are random. This can be explained by that since the tools are mostly very slim compared to other objects, thus the fingers are very close to each other when grasping them. As a result some A features such as $fcon$ contain much information to separate *tool-use* from other tasks.

C. Experiment III: Prediction of Grasp Final Configuration

Since the Bayesian network is a generative model, in addition to evaluate its performance in discrimitive power (i.e. task classification) in the first experiment, we would also like to see how well the model can sucessfully reconstruct any variable given an assigned task. Particularly, we want

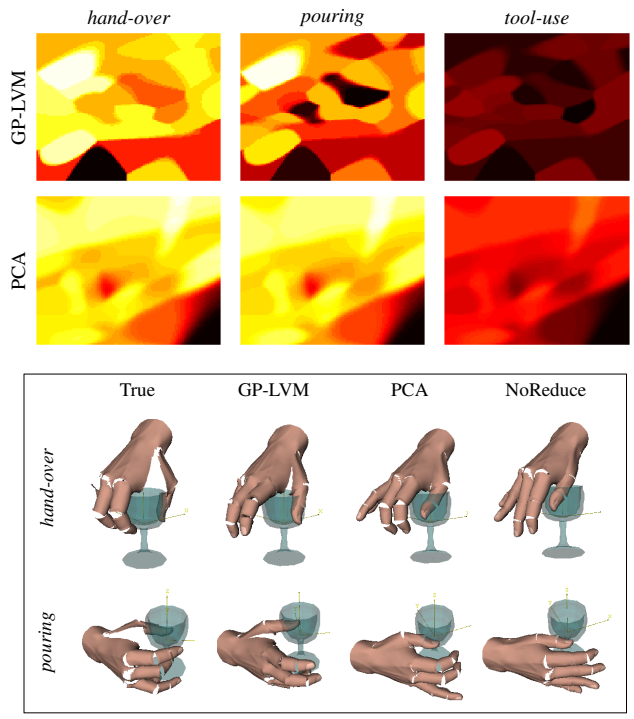


Fig. 5. Experiment III: Top panel shows the likelihood maps of $fcon$ in latent space given tasks and glass's object features $P(fcon|T, O)$; Bottom panel shows prediction of $fcon$ in original space, visualized in GraspIt!.

to show the strength of the proposed discretization scheme GP-LVM on reconstructing the high-dimensional variables. Due to the space limit of the paper, we choose the grasp final configuration $fcon$ as our target variable. But note that the similer effects have been observed on other variables. In this experiment, the goal is to demonstrate that using the discretization scheme GP-LVM, as compared to PCA and NoReduce, we can *i)* better model the constraining effect of the manipulation tasks on $fcon$, and *ii)* predict more intuitive $fcon$ in the original hand configuration space.

To do so, we first obtain the likelihood maps of $fcon$ in the 2D latent space for the GP-LVM and PCA models (see top panel of Fig. 5). The light color of the map indicates that the point has high likelihood for the task. The maps are generated by evenly sampling the posterior distribution $P(\cdot|T, O)$ under the BN model for each task using the same object (glass). This object comes from the test data in order to investigate performance on generalization outside the training data.

The main observations here include: *i)* for both GP-LVM and PCA models, the *tool-use* task has much darker maps, indicating that the glass is clearly not tool-usable; *ii)* when comparing the maps between the two models, we see there are clear differences between *hand-over* and *pouring* tasks for GP-LVM, whereas for PCA the likelihood patterns are almost same. This implies that the GP-LVM model has captured the potential constraining effects of the tasks on the final configuration of the hand. On the contrary, the PCA model is almost 'blind' on the potential task constraints.

As the 2D latent space representation can not give us any intuition about how good is the $fcon$ predicted by the model,

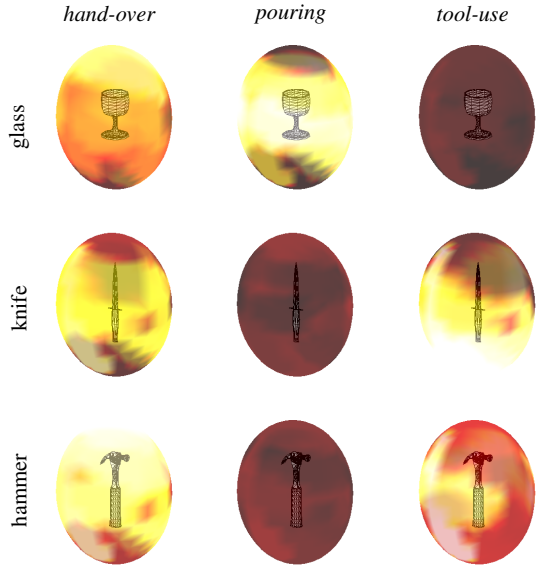


Fig. 6. Experiment IV: Likelihood of Grasp Position Given Tasks and Object Features $P(pos|T, O)$, resulted from the GP-LVM model.

we would like to project the prediction into the original 20D space and visualize it in the simulator. This way, we can also visualize the results from the NoReduce model. The bottom panel in Fig. 5 shows the results with the same object, for the *hand-over* and *pouring* tasks. From the left to right columns, the images represent the ground truth $fcon$, predicted $fcon$ from GP-LVM, PCA and NoReduce models respectively. We can see that the reconstruction of the GP-LVM model is closer to true $fcon$, and importantly the hand configurations are more natural compared to other models. Both the PCA and NoReduce models have unintuitive, even 'impossible' hand configurations that are far away from the true data.

The reason lies on the foundamental differences in the discretization schemes. The GP-LVM model provides a generative discretization scheme, while PCA and NoReduce models do not. As a result, GP-LVM-based BNs can model $P(fcon|T, O)$, where $fcon$ represents the original 20D continuous data, whereas the other models can not. In other words, the proposed GP-LVM-based discretization scheme allows us to construct a *full generative framework* that includes BN, GP-LVM and GMM. This framework is very powerful to model the constraints for robot grasping and manipulation tasks.

D. Experiment IV: Inference on Grasp Position

From the first three experiments, we have shown that the GP-LVM-based discretization scheme outperforms others in both task classification and data reconstruction. The goal of the last experiment is to confirm that GP-LVM model can successfully encode some task constraint on different objects and tasks. Notice that the constraint of a given task is often encoded by a combination of multiple features, e.g. one should not grasp this object from this position pos, in this orientation dir, and with this joint configuration $fcon$. However due to space limit and for the purpose of easier evaluation by the readers, we choose a single

intuitive variable, the grasp position pos, to visualize the task constraint.

Similar to obtaining the likelihood map of $fcon$ in experiment III, we sample 625 grasping positions evenly on an ellipsoid around the object. The size of the ellipsoid is determined by the training data so that the ellipsoid envelops the outer surface of all the grasping positions. As shown in Fig. 6, for each sampled position, the likelihood is obtained given the 3 tasks, and the object features for 3 unknown objects: a glass, a knife and a hammer, i.e. $P(pos|T, O)$.

Fig. 6 shows that the model sucessfully rules out the glass for *tool-use*, and the knife and hammer for *pouring*. For *pouring*, the glass can not be grasped from the top as it will block the opening of the glass; similarly, when using the knife or hammer as a tool, the grasp should avoid the sharp blade or the head of the hammer as functional parts. All the 3 objects afford the *hand-over* task, and the likelihood maps of pos for *hand-over* also capture the intuitive constraints for these objects. Similar results are also observed in PCA model but not in NoReduce model. The reason is revealed by previous task classification results as shown in Fig. 4: while PCA model has good classification rate given the observation on O, A features, the NoReduce model is almost random.

VII. Conclusion

In conclusion, the sparse GP-LVM-based discretization method excels over other methods in learning and inference with Bayesian networks. The compact, efficient data representation allows fast structure learning for BNs that model a large number of variables. And the resulted BN performs significantly better in both task classification and data reconstruction. In addition, since GP-LVM is a generative technique for dimensionality reduction, the model encodes the likelihood of each point in the latent-space, which, when combined with the prediction from BNs, can reproduce much more accurate and intuitive high-dimensional variables such as hand grasp configurations. This presents a major advantage of our proposed discretization method in the field of robotic applications, where many sensory and motor signals are high-dimensional with complex distributions.

In this paper, we only used human hand model to present the discretization techniques for learning Bayesian networks. However, the framework can be generalized to any hands, from which the training data should be generated. The goal of the hand-specific Bayesian network is to allow the task information to be transferred between different hands or embodiments, therefore the goal-directed grasp imitation can be performed [5].

There are some limitations in the current approach that need further research. Firstly, the number of discrete states are manually chosen to satisfy a trade-off between refined data representation and complexity of BNs. In the future, we would also like to learn this hyper parameter automatically from data. Secondly, the inducing points create a sparse model of the full GP, however, there is nothing in the model that encourages the inducing points to be sparse themselves, i.e. less inducing points. Sparseness among the inducing points would reduce the amount of shared explanation, which we believe can lead to better clustering.

We would also like to test our discretization based task constraint model in the real robot platforms where sensorimotor uncertainty is more prominant. We believe this will further exemplify the benifits of using a probabilistic model capable of dealing with uncertainty in real-world applications.

Acknowledgments

This work was supported by EU IST-FP7-IP GRASP, EU IST-FP6-IP-027657 PACO-PLUS, and Swedish Foundation for Strategic Research.

References

[1] S. Kang and K. Ikeuchi, "Toward Automatic Robot Instruction from Perception-Mapping Human Grasps to Manipulator Grasps," *IEEE Transactions on Robotics and Automation*, 1997.

[2] C. L. Nehaniv and K. Dautenhahn, Eds., *Imitation and Social Learning in Robots, Humans, and Animals: Behavioural, Social and Communicative Dimensions*. Cambridge University Press, 2004.

[3] L. Montesano, M. Lopes, A. Bernardino, and J. Santos-Victor, "Learning Object Affordances: From Sensory–Motor Coordination to Imitation," *IEEE Transactions on Robotics*, 2008.

[4] P. Pastor, H. Hoffmann, T. Asfour, and S. Schaal, "Learning and Generalization of Motor Skills by Learning from Demonstration," in *IEEE International Conference on Advanced Robotics*, 2009.

[5] D. Song, K. Huebner, V. Kyrki, and D. Kragic, "Learning Task Constraints for Robot Grasping using Graphical Models," in *IROS*, 2010.

[6] J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, September 1988.

[7] P. Leray and O. Francois, "BNT Structure Learning Package: Documentation and Experiments," Université de Rouen, Tech. Rep., 2006.

[8] L. D. Fu and I. Tsamardinos, "A Comparison of Bayesian Network Learning Algorithms from Continuous Data," in *AMIA*, 2005.

[9] H. Wei, "A Novel Multivariate Discretization Method for Mining Association Rules," in *APCIP*, 2009.

[10] N. D. Lawrence, "Probabilistic Non-linear Principal Component Analysis with Gaussian Process Latent Variable Models," *Journal of Machine Learning Research*, vol. 6, pp. 1783–1816, November 2005.

[11] I. Ebert-Uphoff, "A Probability-Based Approach to Soft Discretization for Bayesian Networks," Georgia Institute of Technology. School of Mechanical Engineering, Tech. Rep., 2009.

[12] O. Colot, P. C. Olivier, and A. El Matouat, "Information Criteria and Abrupt Changes in Probability Laws," in *Signal Processing VII: Theorie and Applications*, 1994, pp. 1855–1858.

[13] T. Cox and M. Cox, *Multidimensional Scaling*, 2001.

[14] J. Tenenbaum, V. Silva, and J. Langford, "A Global Geometric Framework for Nonlinear Dimensionality Reduction," *Science*, vol. 290, no. 5500, pp. 2319–2323, 2000.

[15] K. Weinberger, F. Sha, and L. Saul, "Learning a Kernel Matrix for Nonlinear Dimensionality Reduction," *ICML*, 2004.

[16] M. Tipping and C. Bishop, "Probabilistic Principal Component Analysis," *Journal of the Royal Statistical Society*, 1999.

[17] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning*. The MIT Press, 2005.

[18] M. Titsias, "Variational Learning of Inducing Variables in Sparse Gaussian Processes," in *Artificial Intelligence and Statistics*, 2009.

[19] A. Geiger, R. Urtasun, and T. Darrell, "Rank Priors for Continuous Non-Linear Dimensionality Reduction," in *CVPR*, 2009.

[20] K. Huebner, "BADGr - A Toolbox for Box-based Approximation, Decomposition and GRasping," in *IEEE/RSJ International Conference on Intelligent Robots and Systems: Workshop on Grasp Planning and Task Learning by Imitation*, 2010, **URL:**http://www.csc.kth.se/∼khubner/badgr/.

[21] A. T. Miller and P. K. Allen, "GraspIt! A Versatile Simulator for Robotic Grasping," *IEEE Robotics and Automation Magazine*, 2004.

[22] G. Schwarz, "Estimating the Dimension of a Model," *Annals of Statistics*, vol. 6, no. 2, pp. 461–464, 1978.

[23] C. Chow and C. Liu, "Approximating Discrete Probability Distributions with Dependence Trees," *IEEE Transactions on Information Theory*, vol. 14, no. 3, pp. 462–467, 1968.