Action learning from human demonstrations for personal robots

MASTER OF SCIENCE THESIS

For the degree of Master of Science in Mechanical Engineering at Delft University of Technology

C. G. Rozemuller

December 1, 2013

Faculty of Mechanical, Maritime and Materials Engineering (3mE) · Delft University of Technology
Household robots need to perform tasks specific for the owner. With Learning from demonstration (LfD) a robot can learn new tasks from human demonstrations, without requiring programming skills. This thesis investigates a novel representation of actions that can be learned by using only a 3d camera and an object tracker. The action representation is object-based so it is independent of the morphology of the robot.

The actions are represented using the average and standard deviation of multiple demonstrated trajectories with six degrees of freedom. The standard deviation serves as a weight factor for the required accuracy of the recognized or synthesized trajectory. Three novel methods proposed in this thesis aim to reduce variances in the demonstration that are not specific to the action. First the demonstrations are aligned in time using a novel action signature and a novel time warp algorithm. The time warp algorithm can approximate the alignment of multiple multidimensional signals in quadratic computing time. The third novel technique is a dynamically optimized choice of reference frame so variations in start and end position have little influence on the variance in trajectory.

This method has been tested on a database of five actions repeatedly demonstrated by six subjects. The results show that it is possible to have a 90 percent action recognition rate with only three demonstrations in the database. It is also shown that a robot can use this action representation to synthesize four out of five actions with varying object positions.
# Table of Contents

| Acknowledgements | v |
| Introduction | 1 |
| 1-1 Background | 1 |
| 1-2 Learning from demonstration | 1 |
| 1-3 Examples | 3 |
| 1-4 Classifying the problem | 4 |
| 1-4-1 Framework | 4 |
| 1-4-2 Action complexity | 5 |
| 1-4-3 Tasks | 6 |
| 1-5 Problem statement and proposed approach | 6 |
| 1-6 Structure of the thesis | 7 |
| 2 Related work on learning from demonstration | 9 |
| 2-1 Related work | 9 |
| 2-1-1 Capturing the demonstration | 9 |
| 2-1-2 Generalization | 11 |
| 2-1-3 Task execution | 12 |
| 2-2 Proposed approach | 12 |
| 3 Action representation | 15 |
| 3-1 Preprocessing and normalization | 16 |
| 3-2 Action signatures | 18 |
| 3-3 Multiple action alignment | 19 |
| 3-3-1 Dynamic Time Warping | 19 |
| 3-3-2 Multiple Multidimensional Projected Dynamic Time Warping | 20 |
| 3-4 Reference Frame Optimization | 23 |
# Table of Contents

4 Learning & Recognition 29  
4-1 Learning ........................................ 29  
4-2 Recognition .................................. 30  

5 Synthesis 35  
5-1 Trajectory synthesis .......................... 35  
5-2 Inverse kinematics ............................ 36  
5-3 Advanced synthesis ........................... 37  

6 Experiments & Results 39  
6-1 Pilot recognition experiment .................. 39  
6-2 Experiment set-up .............................. 40  
6-3 Action recognition experiment ............... 42  
   6-3-1 Personal situation ....................... 42  
   6-3-2 General situation ....................... 45  
6-4 Synthesis experiment .......................... 47  

7 Conclusion & Future work 51  
7-1 Conclusion .................................... 51  
7-2 Future work .................................. 52  

A Learning from demonstration in literature 55  
A-1 Surveys ........................................ 55  
A-2 Research groups .............................. 56  
A-3 Capturing the demonstration ................. 58  
   A-3-1 Observation of objects .................. 58  
   A-3-2 Observation of motions .................. 60  
   A-3-3 Observation and user interaction ...... 62  
A-4 Task generalization ......................... 64  
   A-4-1 Logic .................................... 65  
   A-4-2 Statistical ............................... 67  
   A-4-3 Reinforcement learning .................. 67  
A-5 Task execution .............................. 68  

B Data plots 71  

Bibliography 77  

C. G. Rozemuller  
Master of Science Thesis
I would like to express my deep gratitude to my supervisor Dr.ir. Maja Rudinac for her support and the freedom she gave me during the making of this thesis. I would also like to thank the other colleagues in the lab for providing me with useful suggestions and equipment.

Finally would also like to thank my friends and family for their support, motivations and the very fruitful discussions. The cover picture is dedicated to the coffee breaks during which the best ideas have emerged.

Delft, University of Technology
December 1, 2013

Acknowledgements
Chapter 1

Introduction

1-1 Background

General household robots will play an important role in the future. Humanity has always fantasized about machines that can take over any task they want. People want robots to do chores [1] such as cleaning the house, collecting and washing laundry, watching the house, searching for lost keys or do any other task that makes life more comfortable. Bill Gates predicts [2] that these household robots will be a 50 billion dollar industry by 2025 and having one will be just as common as having a computer. This trend already started over a hundred years ago with the first washing machines, automatic dishwashers, vacuum cleaners, lawn mowing robots etc. However a robot that can do anything you want it is still a future scenario.

There is one group of people in particular that can benefit from the development of a general household robot: elderly people. For them a robot that can do household tasks may mean a prolongation of their independent life. Elderly people are the target group for the Delft Robotics team where this research is conducted. We aim to develop affordable personal robots that can be deployed in any domestic or care environment. Figure 1-1 shows the robots developed by this team. The LfD framework developed in this research should be applicable to this robot.

1-2 Learning from demonstration

The ideal household robot should be able to take over any task that the user requires. Therefore everyone should be able to "teach" a robot the desired skills. This rules out every conventional way of programming robots, because they require technical knowledge about the system. The solution is found in a method that is already natural for everyone: teaching and demonstrating. Making a robot learn from human demonstrations will be the topic of this thesis.
Introduction

Figure 1-1: The personal robots developed by the Delft Robotics team. On the left is Robby developed in 2012 and on the right Lea developed in 2013
The field of Learning from Demonstration (LfD)\(^1\) deals with the problem of transferring a policy (task) from a human teacher to a robot learner. Learning from demonstration is a relatively new topic in robotics with publications starting from around 2000. It is a subtopic in the field of supervised learning. It emerged from the topic of Programming by Example (starting from 1980) which focuses on software development based on examples. LfD is different because it is intended for robots that learn human tasks. Because LfD is a relatively new and diverse topic there is not yet a standard framework although many researchers have proposed solutions for parts of the problem.

There are two major challenges that have to be solved in any LfD framework:

**Correspondence and Mapping** A robot is different than its human teacher. For example the human teacher has two hands with five fingers and 22 degrees of freedom whereas the robot may just have a single gripper. How should the robot interpret all these differences? Moreover, it is not possible to perfectly capture every single aspect of the demonstration (forces, temperatures, visual, etc.), so what does describe the essence of the demonstration?

**Generalization** The human demonstrator will only show a limited number of examples of a task, but expects the robot to deal with every variant. For example: the task of pouring a drink may have been demonstrated in the kitchen with a mug and a teapot, but it should also work with a cup and a bottle in the living room.

### 1-3 Examples

Only a few labs in the world already have functional learning from demonstration robots. These prototypes are still very specific and not ready for general use, but nevertheless impressive:

- A Japanese robot can learn language from interaction with its teacher [4]. This robot understands a sentence like 'move the apple around the lemon' after it learned the concepts of 'apple', 'lemon' and 'moving around'.
- There is a robot dog that walks around and learns object structures from conversations with its teacher [5]. After training it could recognize abstract structures such as a column or arch build from different objects. It also learned the concept of flat by rolling a ball over the object.
- A common problem for robot arms is to use the right type of grip to pick things up. The robot made by Steil [6] was able to observe the human grabbing objects and learn the correct technique.
- A robot learned from demonstration how to put items in the fridge, even if the fridge was closed or the intended spot in the fridge was occupied [7, 8].

\(^1\)Also referred to as: Learning by Demonstration (LbD), Programming by Demonstration (PbD), Learning by Experienced Demonstrations, Assembly Plan from Observation, Learning by Showing, Learning by Watching, Learning from Observation, behavioral cloning, imitation and mimicry. [3]

Master of Science Thesis

C. G. Rozemuller
• A German Robot learned to operate a coffee machine to make coffee after observing just three demonstrations [9]. It could also handle new situations where the cup was moved to a new location.

• Figure 1-2 shows a robot that can scoop sugar and wipe the table with unknown obstacles in the way [10]. The robot learned these techniques after analyzing the human demonstrations.

• Recently researchers made a robot that could help assemble furniture after learning from observation [11]

The only thing that all these examples have in common is that they are called learning from demonstration. This shows how diverse the field of LfD is, and how loosely the term is used.

1-4 Classifying the problem

To get a better definition of learning from demonstration literature was consulted. Several subdivisions can be made in the LfD literature that help specify the problem and its solutions. The subdivisions also point out some of the assumptions that have to be made. In this section the most important subdivisions are shown and the choices that have been made are explained.

1-4-1 Framework

There is not a single method to do LfD. Every solution is a mix of different methods, however mostly within the same three step framework. Solutions for all three parts of the framework are required in order to get a functional LfD method. This practical subdivision of methods is used to organize the literature study in chapter 2.
Capturing | Using sensors to capture the essence of the demonstration. This part contains object recognition, capturing motions and including other sources of information.
---|---
Generalization | Formulating the most general description of the task. This part of the framework analyzes the captured data to find underlying rules.
---|---
Execution | Using the general description to execute a task while dealing with changes in the environment.

### 1-4-2 Action complexity

Other than this very practical subdivision there are two subdivisions of action complexity. In a review paper by Krüger [12] the levels of complexity are labeled as: action primitives, actions and activities. The basis for this subdivision is the emerging idea that actions can be described with a grammar just like speech. There are clues that movements and speech are handled by the same part of the brain. The levels of complexity of tasks can be compared to phonemes, words and sentences of speech.

<table>
<thead>
<tr>
<th>Action primitives</th>
<th>The smallest building blocks that form actions. They describe short movements or operations. e.g. twisting of a screwdriver.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actions</td>
<td>A specific composition of action primitives that deals with part of a task, e.g. tightening a screw</td>
</tr>
<tr>
<td>Activities</td>
<td>The combination of actions that solves a task. To combine different tasks into an activity there is usually some logic involved, e.g. Getting a screwdriver from a toolbox and tighten a screw.</td>
</tr>
</tbody>
</table>

In a similar way Steil [6] separates the learning process in three time scales:

<table>
<thead>
<tr>
<th>Situated learning</th>
<th>Rapid learning. With techniques such as imitation, simulation and sensory exploration the system should learn new knowledge from several demonstrations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Local refinement level</td>
<td>Tuning all the subsystems in an on-line manner. Its algorithms can be based on the ideas of statistical learning, e.g. applying skin calibrations for hand recognition.</td>
</tr>
<tr>
<td>Ontogenetic level</td>
<td>The slowest level of learning in which static knowledge and logical rules are updated. This is an off-line process. Learning on this level can also be referred to as datamining. e.g. updating an image recognizer with new knowledge.</td>
</tr>
</tbody>
</table>

The time scales of learning are closely related to the complexity levels of learning. Learning on an Ontogenetic level is required to find the rules (or grammar) of an activity. However for learning an action the lower levels of learning are more suitable.

This research will be focused on designing a framework capable of accurately describing only the first two levels of action complexity. In the example of pouring a drink that would
mean: The reason for the pouring or the relation between the bottle and the cup will not be described by the framework. The goal is finding an accurate and robust description of the pouring motion. Intuitively this level of action complexity is also known as muscle memory. For example: anyone can write their name without involving any logic.

1-4-3 Tasks

Pardowitz [13] proposed another distinction of methods based on the goal of the task.

<table>
<thead>
<tr>
<th>Transport operations</th>
<th>Changing the location of an object, e.g. Placing a cup</th>
</tr>
</thead>
<tbody>
<tr>
<td>Device handling</td>
<td>Changing the state of an object, e.g. Opening a bottle</td>
</tr>
<tr>
<td>Tool handling</td>
<td>The use of one object to change the state of a second object, e.g. pouring the bottle</td>
</tr>
</tbody>
</table>

Knowing the type of task helps selecting appropriate methods. For example: Obstacle avoidance can prevent pushing buttons, but it helps when transporting a cup of water. This work is focused on the third kind of task that has two objects involved: One object that is being operated and a second object that is passively involved.

1-5 Problem statement and proposed approach

The question that will be answered in this thesis is: How can a robot learn a household task from human demonstrations in an unknown household environment, without the drawbacks of current methods?

Current methods to learn from demonstration are not practical when used in a real environment. All of the examples from the state of the art in section 1-3 have one or more limitations that prevent them from being used:

- They require a detailed model of the environment on forehand in order to make generalizations.
- They require physical interaction with the robot to capture the action.
- They require advanced sensory systems that are only available in a specially equipped environment.
- They are not robust against a changing environment.

In this thesis a novel LfD method is proposed that can overcome these limitations. It can be used to learn actions (as defined in the previous section) and it is robust to changes in the environment and natural variations in user demonstrations. In addition it is also user friendly and natural to use.

The proposed action representation can describe a tool handling task by using only the relative trajectories of the two objects involved. This object-based task description has the advantage
that it is independent of the human or robot performing the task. Therefore it avoids the correspondence and mapping problem from section 1-2. This description can also be learned without having a detailed model of the environment.

To avoid physical interaction with the robot, it learns actions using only a common 3d camera. It can learn from observing natural demonstrations in any environment without preparation. It is the first time that a simple 3d camera instead of expensive lab equipment is used to learn actions of this complexity.

The proposed method incorporates three novel techniques to make the task description robust to a changing environment:

1. A six-dimensional action signature to recognize similar parts of actions
2. The fast alignment of multiple action signatures to reduce variance in timing between demonstrations.
3. Optimization of a dynamic reference frame to cope with changing object locations.

The proposed method will be tested with a recognition and synthesis experiment.

1-6 Structure of the thesis

A summary of the literature study that has led to the proposed approach can be found in chapter 2. A detailed description of the proposed approach is given in the second part of this chapter. The comprehensive literature study is added in appendix A.

The way actions are represented with the proposed method is explained in chapter 3. Chapter 4 explains how this representation can be used to learn an actions and how it can be used for recognition. The reproduction of actions from the database on a robot is explained in chapter 5 about synthesis.

Finally the proposed method is tested and discussed with a recognition and synthesis experiment in chapter 6. Chapter 7 concludes this thesis and gives recommendations for future work.
This chapter presents the results of a literature survey for the state of the art LfD methods. In this survey I searched for LfD methods that can be used to learn a task in a household situation. The goal of the literature survey was find a (combination of) methods that can be used to achieve LfD, without the drawbacks mentioned in the problem statement. The resulting method is presented in the second part of this chapter.

2-1 Related work

As could be seen from the examples in section 1-3 the term LfD is widely used for many different ways of learning. A comprehensive overview of the literature on LfD can be found in appendix A. The literature that is relevant for this thesis was narrowed down using the classifications on action complexity, time scale and practicality. The remaining literature presented in this section is divided according to the three parts of a typical LfD framework: Capturing, Generalizing, and Task execution.

2-1-1 Capturing the demonstration

When capturing a demonstration three aspects are considered important in literature: Capturing movement, classifying objects and the use of multiple sources of information.

Capturing motions

Most literature considers movement the most important aspect of an action. As neuroscientist Daniel Wolpert once said: “We have a brain for one reason and one reason only - and that’s to
produce adaptable and complex movements.” There are two ways to capture the movements that are involved in an action: Either by measuring the posture of the person or robot that performs the action, or by measuring the movement of the objects that are involved in the action. The resulting trajectories are respectively in joint-space or in object-space.

The easiest way to capture the motions in a demonstration is by doing the demonstration through the robot’s interface and saving the control inputs. This is called Kinesthetic teaching. It is used in the research of Ye [10] and Schneider [9] to demonstrate the task. For a household robot this is not a natural way of interfacing suitable for daily use. This choice also has a big implication for the way the demonstration is stored. When manipulating a robot arm the motions are stored in joint-space. Kinesthetic teaching makes it more easy to generalize the motion for a specific manipulator, but it requires extra conversion for other manipulators. The alternative is storing the motions in object-space, so it is independent of a kinematic configuration and thus avoiding the mapping problem.

The object-based approach was used successfully by Leonel Rozo [11] to learn assisted furniture assemblage. However they used a special sensor-equipped room to follow objects, which is not available in household situations. Therefore neither of these techniques are suitable to capture a demonstration in a household situation. A feasible alternative for this sensor-equipped room is to use a camera-based object tracking.

Conventional object tracking such as TLD [14] can track objects in a 2D camera image. It was shown by Ye and Schneider that object orientation and position are very valuable for recognition, and crucial for synthesis. Therefore 3D object tracking is required. A 3D tracker that can do this is part of the point cloud library [15] and works with a Kinect 3D camera. This system can accurately output the position and orientation of objects on the scene that are used for the demonstration.

This thesis proposes to represent actions in object-space, using camera-based object tracking. This is both an advantageous to avoid a mapping problem and for a natural user experience.

After finding an object-space description Krüger [16] shows that the motion can be segmented into action primitives. The PHMM (Parameterized Hidden Markov Models) technique proposed by him is feasible to describe and segment motions obtained from an object tracker. The segmentation helps to more accurately describe actions independent of timing or direction. This is a great feature to make a task description more robust. Therefore segmentation, alignment, and a description independent of direction will be used to make the proposed approach robust.

An alternative method to segment a motion is given by Dillmann [17] who uses the speed of the hand to segment key-moments in a demonstration. A third alternative by Wu [18], using curvature, is more accurate for alignment at the cost of using higher derivatives. Neither of these three methods can align actions with six degrees of freedom (position and orientation). Therefore a novel method is required that will be based on these three methods.

Classifying objects

Ideally LfD system should be able to identify objects that it has seen before or recognize unknown objects that have similar affordances. That way the system can learn which objects are needed to complete the task. Object recognition is already possible with the current...
version of the robot [19], however affordance recognition is not. This topic has not received much attention in LfD research. Only SOINN by Hasegawa [20, 4, 21, 22] was used to classify unknown objects in a LfD setting. Other techniques based on pose recognition [23], object shape [24] or virtual object usage [25] seem to work just as well or even better to recognize affordance.

There is not yet a well-established method for object affordance recognition in LfD. The development of this part is outside the scope of this research. The objects involved are therefore labeled by the user.

**Multiple sources of information**

A lot of LfD research focuses on using multiple sources of information during the demonstration. Speech has proven itself as a good and natural way of giving feedback or correcting the system and can be very beneficial for quick learning. It has been used to learn the users intention [26, 27], to get extra instructions [28, 29, 30, 31], to get feedback [32] or to get advice [33, 34, 35, 36]. Also shared attention, using gaze or pointing, is used a lot to assist LfD [6, 20, 21, 5].

In most examples from literature the extra source of information is used as addition to a learning system that can also work without it. Implementing a second source of information is outside the scope of this research and will be a recommendation for future work.

**2-1-2 Generalization**

In literature two major options can be found to generalize the demonstrations into a policy. With symbolic logic the rules that describe a policy can be found. Alternatively there are methods that use a statistical description as a the policy.

Most of the generalization methods in literature are based on symbolic problem solving [37, 21, 17, 38, 28]. Symbolic logic can be used to generalize tasks that require multiple steps and can be done in multiple orders. These methods are used to generalize tasks on long time scales, but they lack the flexibility to describe the details on the situated learning level. Jäkels method of automatic constraint generation [7, 39, 8] is more suitable to describe tasks with more detail. It however requires a very detailed model of every object in the scene, thus making it unfeasible for daily life usage. The requirement of detailed models or lack of detail on a fast time scale makes the symbolic method not feasible to learn household actions.

The statistical distribution based methods from Schneider [9] and Ye [10] are a detailed enough to describe tasks on a fast time scale. They are able to generalize all important movements relative to specific objects. It is not suitable for one-shot learning so multiple demonstrations are required. The method used by Ye [10] has been successfully combined with a path planner to avoid collisions and has therefore a slight practical advantage over Schneiders method. Both techniques have been applied with success in LfD, however using a joint-space movement description.

In this thesis a novel object-space method is proposed that also uses a statistical distribution as task description.
2-1-3 Task execution

Robust task execution has not received much attention in LfD. Only a few frameworks can deal with obstacles or other unexpected conditions that were not demonstrated. The robust task description by Kruger [16] can be used to deal with new conditions. Also a task description that is relative to the objects (mentioned in the previous section) deals with objects that move around. This thesis proposes to use a novel reference frame optimization (RFO) to describe tasks relative to objects.

To make a task even more robust, extra skills, such as obstacle avoidance, have to be added to the framework. As mentioned in the introduction, this gives extra problems because each task requires different skills. For example with an object avoider the robot can never click a button. Pardowitz’s task differentiation can help to select skills that are suitable for the task. This will not be the focus of this research. In this research it is assumed that there are no obstacles that require extra skills.

To execute an object-based task description the robot’s end effector has to pick up the object and follow a generated trajectory. Reinforcement learning has been used to find an efficient robot-specific task execution, but that is outside of the scope for this research. A Jacobian based path planning algorithm [40] will be used to execute tasks on a robot.

2-2 Proposed approach

In this section a novel approach for LfD is presented that is based on the conclusions in the previous section. The block scheme in figure 2-1 shows the steps in this approach and the influence they have on each other.

Learning new actions

1. To learn, the system first observes multiple demonstrations using an object tracker. The actions are demonstrated in a natural environment by human subjects. The demonstrators are instructed to show each action with varying start and end configurations to make the description robust. The subjects have to label their actions and objects involved.

2. Secondly a novel Action Signature is formed for each of the demonstrated trajectories of an action.

3. This action signature is used in the third step to align trajectories in time, with a novel algorithm. An aligned average signature vector is produced for each action that is used in recognition to align an unknown demonstration with. With this step the description is made more robust to changes in timing.

4. In the fourth step the trajectories are made independent of their start and end configuration, by a Reference Frame Optimization. This makes the description robust to changes in object configuration. The optimal parameters after the optimization are used to recognize and synthesize this action.

5. Finally, with variance analysis, a statistic distribution is used to describe the aligned and transformed trajectories.
Recognizing actions

To recognize an unknown demonstration, it is compared to every action in the database until the closest match is found. For every comparison the unknown demonstration goes through the same process as when learning a new action. However the parameters that were found during the learning of each action are applied to transform the unknown trajectory. Finally the unknown transformed trajectory is compared to the statistical descriptions of the actions in the database. It is classified as the action it is most similar to.

Synthesis

For synthesis the average action trajectory is transformed to start and stop at a position of choice. This results in a trajectory that can be tracked by a robot.

The following chapters are structured according to this approach. Chapter 3 is about how an action is represented in a robust way. Chapter 4 describes how this representation will be used for learning and recognition. Chapter 5 describes how it can be used for action synthesis. A recognition and synthesis experiment is used to test the learning performance of this new method in chapter 6.
Figure 2-1: A block diagram of the proposed approach with the steps to learn, recognize and synthesize actions.
Chapter 3

Action representation

The action representation presented in this thesis is a description of the action that contains all the necessary information to recognize or synthesize it. Based on the ideas of Schneider [9] and Ye [10] this representation contains the average trajectory and variance from multiple demonstrations. However in this thesis a new object-space approach is used instead of a joint-space approach. The steps in this chapter make the observed trajectories independent of changes in timing and object location, so the trajectories can be used for statistical analysis in the next chapter.

A demonstrated action that is observed by the system contains more information than just the action. Most of this information is not specific for the action and will make learning and recognition harder. The trajectories are made more distinguishable from other actions by removing variances that are not caused by the choice of action. There are two major causes of variances that are not action specific: Inconsequent timing and different start and end situations.

- Inconsequent timing is caused by the human demonstrator that can do parts of the demonstration faster or slower than other parts. This effect will be reduced by aligning parts of the demonstration that are the same.

- The objects that are involved in the demonstration will have arbitrary start and end positions. A big part of the demonstration is spend on moving the objects towards a position where the actual action starts.

The method presented in this article is for learning and recognizing ‘tool handling’ actions that involve two objects. Before the demonstration starts, the human subject indicates which action will be demonstrated and which two objects are involved. The trajectories of these two objects are captured in six degrees of freedom using a 3D-camera and object tracker (This method will be explained further in chapter 6). This is repeated until a desired number of trajectories from the same action are in the database. The trajectories are then processed and combined into an action representation. The processing takes four steps. First the raw data
is pre-processed and normalized as explained in section 3-1. Secondly a novel action signature of the trajectories is produced according to section 3-2. In section 3-3 the demonstrations are aligned using the signature to remove variance in timing. Lastly variations in start and end configuration are reduced with a coordinate optimization in section 3-4. Appendix B shows plots of trajectory data being processed according to the steps in this chapter.

### 3-1 Preprocessing and normalization

The raw data from an object tracker needs to be formatted and normalized so it is practical to use in the next steps. The raw data consists of the position and orientation of the two objects involved in the action.

The position and orientation of each object at each time step are stored separately in variables $P(t)$ and $O(t)$. Position $P^{i,a}(t)$ of object $a$ in demonstration $i$ is described using Cartesian coordinates as in equation 3-1. Orientation $O^{i,a}(t)$ of the same object is described by the four elements $q_1 \ldots q_4$ of an unit quaternion as in equation 3-2. The choice for unit quaternions, instead of the more typical Euler angles, was made because they do not contain singularities. There is a total of $M$ different demonstrations indicated by index $i$ and a total of two objects indicated by $a$.

\[
P^{i,a}(t) = \begin{pmatrix} x^{i,a}(t) \\ y^{i,a}(t) \\ z^{i,a}(t) \end{pmatrix} \quad i \in [1, M], a \in [1, 2] \tag{3-1}
\]

\[
O^{i,a}(t) = \begin{pmatrix} q_1^{i,a}(t) \\ q_2^{i,a}(t) \\ q_3^{i,a}(t) \\ q_4^{i,a}(t) \end{pmatrix} \quad i \in [1, M], a \in [1, 2] \tag{3-2}
\]

It is assumed that the absolute position and orientation of these objects is not important for the action representation, because they are based on an arbitrary start situation. Therefore trajectories are expressed relative to each other as in equations 3-3 and 3-4. In equation 3-4 $\bar{O}$ is the quaternion inverse of $O$ and $\otimes$ is the quaternion multiplication.

\[
P^{i}(t) = P^{i,1}(t) - P^{i,2}(t) \tag{3-3}
\]

\[
O^{i}(t) = \bar{O}^{i,1}(t) \otimes O^{i,2}(t) \tag{3-4}
\]

The relative speed and relative rotational speed are calculated according to equation 3-5 and 3-6. The Euclidean norm $\| \cdot \|$ was used to combine the three velocity elements into one value for speed.

C. G. Rozemuller

Master of Science Thesis
\[
\begin{align*}
\mathbf{v}^i(t) &= \left\| \begin{pmatrix} v_x \\ v_y \\ v_z \end{pmatrix} \right\| = \left\| \frac{dP^i(t)}{dt} \right\| \\
\mathbf{\omega}^i(t) &= \left\| \begin{pmatrix} \omega_x \\ \omega_y \\ \omega_z \end{pmatrix} \right\| = \left\| 2 \frac{d\mathbf{O}(t)}{dt} \otimes \mathbf{\dot{O}}(t) \right\|
\end{align*}
\]

(3-5) \hspace{10cm} (3-6)

The start of the action is defined as the first moment when one of the objects moves, and the end as the last moment one of them moves. The parts of the trajectory before and after these moments are filtered out using equations 3-7 and 3-8. \( v_t \) and \( \omega_t \) are speed thresholds that should be set manually to make movements distinguishable from noise.

\[
\begin{align*}
t_{\text{start}} &= \min \left\{ t | v^i(t) > v_t \lor \omega^i(t) > \omega_t \right\} \\
t_{\text{end}} &= \max \left\{ t | v^i(t) > v_t \lor \omega^i(t) > \omega_t \right\}
\end{align*}
\]

(3-7) \hspace{10cm} (3-8)

The average length of the trajectory \( \bar{t}_{\text{end}} \) is a property of the action.

\[
\bar{t}_{\text{end}} = \text{mean}(t_{\text{end}}^1, t_{\text{end}}^2, \ldots, t_{\text{end}}^M)
\]

(3-9)

Finally we will normalize the length of \( P^i(t) \) and \( O^i(t) \) in time so all demonstrations contain the same number of time samples \( N \). This is convenient for the calculation of the alignment in the next subsection. We have chosen \( N = 100 \) so the demonstrations with a maximal duration of 10 seconds are sampled at least 10Hz. Notation \( P^i_n \) is used to indicate position \( P \) in demonstration \( i \) at sample \( n \)

\[
\begin{align*}
t^i_n &= \frac{n \ast t^i_{\text{end}}}{N} \quad \text{for } n = 1, 2 \ldots N \\
P^i_n &= P^i(t^i_n) \\
O^i_n &= O^i(t^i_n) \\
v^i_n &= v^i(t^i_n) \\
\omega^i_n &= \omega^i(t^i_n)
\end{align*}
\]

(3-10) \hspace{10cm} (3-11) \hspace{10cm} (3-12) \hspace{10cm} (3-13) \hspace{10cm} (3-14)

The normalized trajectory can now be used to make action signatures. Although there is noise in the data no smoothing or filtering was applied. This is because later the average and standard deviation of this data are used for recognition and synthesis and that also smooths the data.
3-2 Action signatures

The position and orientation at each time step will form the basis for a statistical description of the trajectory. By aligning the trajectories it is made sure each point in time represents the same moment of the demonstration. This way the details of the trajectory will be preserved. For example during one demonstration the subject can be pouring at n=30, but during a second demonstration the subject is still moving the bottle.

However the trajectories of the same action can look completely different in different demonstrations, even when they are aligned. This is because the objects are located differently per demonstration, which makes the trajectories different. Alignment methods search for parts of the trajectory that look similar and will therefore not be able to align these trajectories directly. Therefore a different signal is derived from the trajectories that will look the same and can be used to find the correct alignment. This is called an action signature.

According to the theory of Krüger [16] the demonstrations still contains the same sequence of action primitives although they look different. Figure 3-1 shows an intuitive example that indicates four action primitives that together form the action of pouring. An action signature is needed to distinguish these different parts of the trajectory. An action signature is a signal with peaks at the moments when the trajectory changes to the next part. Indicators of a transition can be changes in speed, full-stops or corners.

These signatures have been described by Wu [18] for three dimensional trajectories using curvature, torsion and their derivatives. The downside of curvature and torsion is that they requires higher order derivatives of the trajectory, making it very sensitive for noise. Also this signature does not take orientation into account.
For those reasons a new signature is proposed. This signature should be independent of the direction of the trajectory (pouring from the left is the same as pouring from the right) and also independent of the length of the trajectory (writing a small letter is the same as writing a big letter). Based on those assumptions the proposed signature vector $S_n$ will contain the normalized speed $\bar{v}_n$ and normalized rotational speed $\bar{\omega}_n$.

\begin{align*}
\bar{v}_n^i &= \frac{v_n^i}{\sum_{n=1}^{100} v_n^i} \\
\bar{\omega}_n^i &= \frac{\omega_n^i}{\sum_{n=1}^{100} \omega_n^i} \\
S_n^i &= \{\bar{v}_n^i, \bar{\omega}_n^i\}
\end{align*}

(3-15) (3-16) (3-17)

Speed was chosen because it is independent of the direction. It is normalized by the trajectory length to make it independent of the magnitude of the motion. The normalization also puts the values for translation and rotation in the same numerical range to avoid a bias during the Dynamic Time Warping (DTW) algorithm.

### 3-3 Multiple action alignment

To reduce variance caused by inconsequent timing the trajectories need to be aligned in time according to their signatures. This is not straight forward because conventional methods of alignment cannot align more than two sequences. To overcome this problem a novel extension to Dynamic Time Warping is proposed that makes alignment of more than two signatures possible.

Dynamic Time Warping (DTW) is an algorithm that finds the points of two sequences that need to be aligned in order to get the best matching solution. DTW can only be used to align two signals relative to each other. To align multiple signals a novel algorithm called Multiple Multidimensional Projected Dynamic Time Warping (MMP-DTW) is proposed. There is an existing method \[41\] that can also align more than two samples, but it has the problem of exponential computational complexity, making it unfeasible for aligning multiple signatures. MMP-DTW has only quadratic computational complexity and is therefore suited to align large datasets.

The MMP-DTW method consists of two steps: First conventional DTW is used to align all combinations of two signals relative to each other. Second, these partial solutions are projected on to a universal coordinate system so they may be compared and a global solution can be approximated. In the next sections DTW and MMP-DTW are explained in detail.

### 3-3-1 Dynamic Time Warping

Dynamic Time Warping gives a solution for the alignment of two signals. It is mainly used in speech recognition to cope with different talking speeds, but is also used to recognize speakers, signatures, shapes, and many other things. It computes which points of the original signals
should be at the same point in time, but it does not specify at which point in time. DTW finds a solution by shifting the points in the signals in time to minimize their distance.

DTW works in two steps: First a cumulative distance matrix $\gamma$ is created from two signals, or in this case two signatures $S^i$ and $S^j$. The direction of the minimum distance for all possible combinations of $S^i$ and $S^j$ is stored in a matrix $A$.

**Algorithm 1** Calculate cumulative distance and action matrix

```plaintext
initialize $A$ as an empty matrix with range $[1 \ N] \times [1 \ N]$
initialize $\gamma$ as a matrix with range $[0 \ N] \times [0 \ N]$ and value $\infty$

$\gamma[0,0] = 0$

for $n = 1$ to $N$ do
    for $m = 1$ to $N$ do
        if $\gamma[m-1,n-1] < \gamma[m-1,n] \ or \ \gamma[m,n-1]$ then
            min ← $\gamma[m-1,n-1]$
            $A ← 0$
        else if $\gamma[m-1,n] < \gamma[m,n-1]$ then
            min ← $\gamma[m-1,n]$
            $A ← 1$
        else
            min ← $\gamma[m,n-1]$
            $A ← -1$
        end if
        distance ← $\|S^i[m] - S^j[n]\|$
        $\gamma ←$ distance + min
    end for
end for
```

Secondly in algorithm 2 the shortest route is traced back by following the directions of minimal distance stored in $A$ starting at the end of the trajectories.

After algorithm 2 the indices of the closest matching alignment of signatures $S^i$ and $S^j$ are stored in matrix $K^{i,j}$ (For readability this upper indexing is removed until required). The length of $K$ can vary from $N$ to $2N$. This solution does not specify at which point in time the alignment should take place and is only for two signatures. Therefore DTW cannot directly be used to find the optimal alignment of all the signatures at once.

The exact solution to align multiple samples could be found by making algorithms 1 and 2 more complex so they operate in as many dimensions as there are signals (as is done in [41]). However this would also increase the calculation time exponentially with the number of signals. Aligning ten samples would already require days of calculation, so this method is unfeasible for household robots. Therefore a novel method proposed in this thesis, called MMPDTW, is used that can approximate a solution in quadratic computing time.

### 3-3-2 Multiple Multidimensional Projected Dynamic Time Warping

Multiple Multidimensional Projected Dynamic Time Warping (MMPDTW) is a novel algorithm to align multiple multidimensional signals. It works in three steps: First every possible
Algorithm 2 Trace back the shortest route

\[
\begin{align*}
I_1 & \leftarrow N \\
I_2 & \leftarrow N \\
i & \leftarrow 0 \\
KInv[1,:) & \leftarrow \{N, N\} \\
\text{while } I_1 > 0 \text{ and } I_2 > 0 & \text{ do} \\
i & \leftarrow i + 1 \\
\text{if } A[I_1, I_2] = 0 & \text{ then} \\
I_1 & \leftarrow I_1 - 1 \\
I_2 & \leftarrow I_2 - 1 \\
\text{else if } A[I_1, I_2] = 1 & \text{ then} \\
I_1 & \leftarrow I_1 - 1 \\
\text{else} \\
I_2 & \leftarrow I_2 - 1 \\
\text{end if} \\
KInv[i,:) & \leftarrow \{I_1, I_2\} \\
\text{end while} \\
\text{PathLength} & \leftarrow i \\
\text{for } i = 1 \text{ to PathLength} & \text{ do} \\
K[i,:] & \leftarrow KInv[\text{PathLength}+1-i,:] \\
\text{end for}
\end{align*}
\]

A combination of signature vectors is aligned using normal DTW. Then these solutions are expressed with a variable that quantifies how far the signatures have to be shifted instead of using indices. This variable is called 'time shift'. Lastly the global solution is approximated from all time shifts.

Every solution \(K\) from the DTW algorithm is a \(P \times 2\) matrix containing all combinations of two indices that should be aligned. For every two aligned points a moment in (normalized) time \(n_p\) is chosen where the alignment should take place. \(n_p\) is defined to be centered between the two points of solution \(k_p\), so it is not biased towards one of the signals.

\[
n_p = \frac{K_{1,p} + K_{2,p}}{2} \quad (3-18)
\]

Using algorithm 3 the extra entries of \(K\) are removed so it will have normalized length \(N\). The algorithm works by only keeping the entries of \(K_p\) for which round\((n_p)\) is unique. With this same algorithm also the extra entries of \(n_p\) can be removed, however the result will be that the remaining values of \(n_p\) are the numbers 1 to \(N\). Therefore normalized index \(n\) can be used again instead of \(n_p\).

Then solutions \(k_n\) of DTW are then expressed as a time-shift \(r_n\) that represents how much the original points have to be delayed or progressed in order to be aligned at \(n\). Figure 3-2 shows an example of this time-shift. Each solution \(K^{i,j}\) is specific for the alignment of signature \(S^i\) with \(S^j\) and so are the time-shifts.
The time-shifts can be found for each pair of signatures. This information combined can be used to approximate the best timing for global alignment. e.g. If a signal needs to be delayed to be aligned with most of the other signals then for a global solution it will also be delayed. The global solution for each signature can be approximated by averaging the all the time-shifts $r_{i \rightarrow n}^i$ that are found for each signature $S_i$.

$$
\hat{r}_n^i = \text{round} \left( \frac{\sum_{j=1}^{M} r_{n}^{i \rightarrow j}}{M} \right) \quad \text{for } n = 1, 2, \ldots, N \text{ and } i = 1, 2, \ldots, M \quad (3-21)
$$

The aligned demonstrations are given by equations 3-22 and 3-23 when applying the time shifts.

$$
\hat{P}_n^i = P_{n}^{i+n} \quad (3-22)
$$
$$
\hat{O}_n^i = O_{n}^{i+n} \quad \text{for } n = 1, 2, \ldots, N \text{ and } i = 1, 2, \ldots, M \quad (3-23)
$$
**Algorithm 3** normalizing $K_p$

```
LastValue ← 0
for $i = 1$ to $P$ do
    Value ← round$(n[p])$
    if Value = LastValue then
        remove $k[p]$ from $K$
    else
        LastValue ← Value
    end if
end for
```

All the trajectories are now aligned so the variances caused by inconsequent timing are reduced.

### 3-4 Reference Frame Optimization

In the current formulation the objects are expressed relatively to each other. This results in high variances at the start and end of the trajectories, because those positions are chosen arbitrary by the demonstrator. These variances are not specific for the action and should be filtered out for a more detailed action representation.

To reduce the variance caused by the arbitrary start and end positions a novel method called Reference Frame Optimization (RFO) is proposed. RFO optimizes the choice of reference frame at each point in time. Figure 3-3 shows the effect of using different reference frames with a 2d example. Three possible reference frames have been used:

- The first reference frame is rotated and scaled so all trajectories start at the same location. Most trajectories start with a converging motion to bring the objects closer to each other before the action specific motion starts. Therefore the beginning of many action trajectories will lie close together when this reference frame is used. Because the start of the actions can be at different distances from the object also scaling of the trajectory is applied so the trajectories start at the same position on the x-axis.

- The second frame is the original reference frame that is located in the center of the passive object$^1$.

- The last reference frame is rotated so the end of the trajectories are at the same location, but it is otherwise the same as the first reference frame. This frame can describe the diverging trajectory at the end of the action when objects are placed in an arbitrary location.

According to the theory of action primitives [16] it assumed that every action primitive operates in one specific reference frame. In the previous section the action primitives that

---

$^1$This method can be extended with more object frames when multiple objects are involved in the action. This algorithm would then automatically describe the trajectory relative to the most relevant object, and switch when it is optimal.

Master of Science Thesis

C. G. Rozemuller
Figure 3-3: An example of three reference frames used in Reference Frame optimization and the original trajectory. The four trajectories in this example represent the 2d action of 'moving over a block'. The gray parts indicate where the variance between the trajectories is not optimal because of the choice of reference frame. It can be seen that the start of this action can be most accurately described when the start positions are aligned. The middle when the trajectories are aligned with the second object, and the end when the ends are aligned. Note that the object frame is the same as the original situation.
form the actions have been aligned. Therefore also the moment when reference frames switch should already be aligned.

To find the best moment to switch frames an optimization should be performed that minimizes the total variance in the demonstrations. This process is very computationally intensive so in practice only three variables are optimized:

- The moment $\tau_1$ the start frame switches to the object frame.
- The moment $\tau_2$ the object frame switches to the end frame.
- The duration $d\tau$ of the switch

For the optimization a pattern search algorithm [42] was used. This method systematically tries every combination of variables (within the given limits), but only increases the search accuracy near a minimum.

The switching moments that are found using this method are considered a property of the action. They will not only be used to represent actions robustly, but they are also applied to actions that have to be recognized or synthesized. This will be explained in chapters 4 and 5.

To select the appropriate reference frame they are given a weighing factor $W$ according to the switching moments $\tau_1$, $\tau_2$ and switch duration $d\tau$.

\[
W_{1n} = \begin{cases} 
1 & n \leq \tau_1 \\
0 & n > \tau_1 
\end{cases} \quad (3-24)
\]

\[
W_{2n} = \begin{cases} 
0 & n \leq \tau_1 \\
1 & \tau_1 < n \leq \tau_2 \\
0 & n > \tau_2 
\end{cases} \quad (3-25)
\]

\[
W_{3n} = \begin{cases} 
0 & n \leq \tau_2 \\
1 & n > \tau_2 
\end{cases} \quad (3-26)
\]

For a smooth transition a moving average filter with width $d\tau$ is applied and $[W_{1n}W_{2n}W_{3n}]$ is normalized again after smoothing. Figure 3-4 shows a typical combination of weight factors.

Switching to a specific reference frame can be accomplished by multiplying the trajectory with a rotation matrix and a scalar. The rotation matrix rotates the trajectory so the points of choice are aligned and the scalar corrects for different distances. The rotation matrix is build from three perpendicular unit vectors.

1. First the $x'$-axis must be in the direction of choice using weighing factor $W$. This is a linear combination of a vector pointing to the start of the trajectory, pointing straight, or pointing to the end of the trajectory. The weight and every direction vector are normalized, so the resulting $x'$ vector is a unit vector.

\[
x'^i_n = W_{1n} \frac{\hat{P}_i}{||P_i||} + W_{2n} \begin{pmatrix} 1 \\
0 \\
0 \end{pmatrix} + W_{3n} \frac{\hat{P}_N}{||P_N||} \quad (3-27)
\]
2. Secondly the \( y' \)-axis is derived to be in the original xy-plane. It must be perpendicular to both the original z-axis and new \( x' \)-axis so the cross product is applied and it is normalized.

\[
y^{ri}_n = \frac{x^{ri}_n \times \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}}{\|x^{ri}_n \times \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}\|}
\] (3-28)

3. Thirdly the \( z' \) direction is set perpendicular to the \( z'y' \)-plane by taking the cross product. No normalization is required because both vectors are perpendicular unit vectors.

\[
z^{ri}_n = x^{ri}_n \times y^{ri}_n
\] (3-29)

4. Lastly the complete rotation matrix is composed from these three unit vectors

\[
R^{ri}_n = [x^{ri}_n y^{ri}_n z^{ri}_n]
\] (3-30)

These four steps are repeated for every demonstrated trajectory \( i \) and every step in normalized time \( n \)

The scaling of the transformed trajectories is given by equation 3-31 based on a linear combination of the distance to the start, 1 and distance to the end.

\[
s^i_n = W1_n \ast \|\hat{P}^i_1\| + W2_n \ast 1 + W3_n \ast \|\hat{P}^i_N\|
\] (3-31)

Lastly the new reference frames are applied to correct the original trajectory. The rotation of the trajectory is corrected by multiplying it with the inverse of rotation matrix \( R^{ri}_n \) and the scale is corrected by division by \( s^i_n \). Because the orientation of the trajectory is given as a quaternion the rotation matrix is also converted to a quaternion indicated by operation \( Q(\bullet) \)

---

C. G. Rozemuller  
Master of Science Thesis
\[
\begin{align*}
\hat{P}_n^i &= R_n^{-1} \frac{s_n}{\hat{p}_n^i} \\
\hat{O}_n^i &= Q(R_n^{-1}) \otimes (\hat{O}_n^i)
\end{align*}
\] (3-32) (3-33)

The steps from equation 3-24 until 3-33 are applied for every combination of \(\tau_1, \tau_2\) and \(d\tau\) that is supplied by the optimization algorithm and the variance of the resulting trajectories is evaluated.

\[
\min_{\tau_1, \tau_2, d\tau} \sigma(\hat{P}^{\tau_1, \tau_2, d\tau}, \hat{O}^{\tau_1, \tau_2, d\tau})
\] (3-34)

The trajectories with the lowest variance are used for recognition and synthesis, and the associated values for \(\tau_1, \tau_2\) and \(d\tau\) are stored as a property of the action. The calculation of the variance is explained in the next chapter.

After the steps in this chapter the trajectories of an action are aligned in time and made independent of the start and end position. This generalized description of the demonstrations is therefore very specific for the action. The corrections to the trajectory that were applied in this chapter can also be applied to unknown trajectories to recognize them despite of arbitrary choice of timing and object location. The inverse of these corrections can be applied to synthesize an action with a specific timing and object location.
Chapter 4

Learning & Recognition

The demonstrated trajectories are now aligned and transformed to be independent of the start and end position. The collection of trajectories in the database can now be generalized with a statistical description. This description makes it suitable for recognition and synthesis.

4-1 Learning

The demonstrated trajectories are used to make a statistical description of every action. The variations that are left in the trajectories are all possible ways of doing the same action. It is assumed that these variations are normally distributed around an average trajectory. The average trajectory is therefore most likely to be observed in case of a recurrence of the same action. The standard deviation is weight factor of how much difference is allowed from this average before it becomes a different action. This principle is used to recognize and synthesize actions.

The average position and orientation of the objects for each point in time are estimated with equations 4-1 and 4-2. This average is used for the synthesis of actions. The Markley average $\mathcal{M}(\cdot)$ is used to find the average orientation. This operation produces a unit quaternion that is the closest to all the input quaternions. The implementation of this function is given in the article written by Markley [43]

\[
\bar{P}_n = \frac{\sum_{i=1}^{M} \bar{P}^i_n}{M} \quad (4-1)
\]
\[
\bar{O}_n = \mathcal{M}(\bar{O}^1_n, \bar{O}^2_n, \ldots, \bar{O}^M_n) \quad (4-2)
\]

This is the average trajectory that is used for synthesis of this action. For recognition a different average and standard deviation is used. The average and deviation will be described using a virtual end effector that combines variances in position and orientation. The virtual end effector is a point with a fixed offset from the tracked object. The result will be variance in
three dimensions. The offset of the end effector acts as a weighing factor between orientation and position variances. In our case we choose the end effector to be a point at offset $= [0; 0; 0.15]$ from the object origin. The length of 0.15 is the typical length of the objects that are used in the demonstration.

A virtual end effector was used instead of separate variances for position and orientation because of two reasons: It gives a understandable weighing factor between position and orientation errors when they are inevitably combined to a single number. Secondly the theory of quaternion distributions is not straight forward and outside the scope of this thesis.

The position, and mean and standard deviation the position of the virtual end effector $V$ are given by equations 4-3, 4-4 and 4-5. Operator $R(\bullet)$ is used to indicate the use of a rotation matrix instead of a quaternion.

\[
\begin{align*}
V_n^i &= \tilde{P}_n^i + R(\tilde{Q}_n^i) * \text{offset} \\
\bar{V}_n &= \frac{\sum_{i=1}^{M} V_n^i}{M} \\
\sigma_n &= \text{std}(V_n^1, V_n^2, \ldots, V_n^M)
\end{align*}
\]

Also an average signature vector for this action is produced by equation 4-6, which will be used for recognition.

\[
\bar{S}_n = \frac{\sum_{i=1}^{M} S_n^i}{M}
\]

The distribution of the virtual end effector together with the mean signature and the reference frame switching properties $\tau$ (as defined in section 3-4) describe the action. The database contains these values for all actions that have been demonstrated. Also the original trajectories are stored to be able to extend the database with new trajectories.

### 4-2 Recognition

The most distinctive part of an action is the part with the lowest variance or highest accuracy. New trajectories of that action contain the same distinctive parts of low variance. This property can be used for action recognition. The distance between the average action distribution and the new trajectory is a measure of recognition. Figure 4-1 shows an example of a generalized trajectory and its standard deviation.

Before a recurring action can be recognized the following steps need to be executed:

1. First its trajectory is preprocessed and normalized according to all the steps in section 3-1
2. Secondly a signature vector $S_{\text{new}}$ is created from this trajectory, as in section 3-2
Figure 4-1: Plot of the mean (black line) and standard deviation (blue area) of the virtual end effector. This specific trajectory is for the action of pouring source with a circular motion with 30 demonstrations in the database (see chapter 6 for more details). The moment where the reference frame is switched at \( n = 10 \) and \( n = 92 \) can clearly be distinguished by a higher standard deviation. The rest of the blue area is close to the average. Therefore it serves as a very specific blueprint for the recognition of this action.
3. Then the trajectory is aligned with every action in the database, using the new signature \( S^{\text{new}} \) and the mean signatures \( \bar{S} \). The alignment is done with the normal DTW algorithm from section 3-3-1. Only the new trajectory is shifted in time to get the best match with trajectories in the database.

From this step there are multiple versions of the new trajectory, one for each action in the database. Later it is tested which version is best matching the database.

4. Now the aligned trajectories undergo the same reference frame transformation as their database counterparts. The transformation from equations 3-24 to 3-33 is applied using the values for \( \tau \) that are in the database.

5. Finally for each new trajectory a virtual end effector \( V^{\text{new}} \) is produced with equation 4-3.

For every action in the database the total absolute error between \( V^{\text{new}} \) and \( \bar{V} \) is calculated and normalized with the standard deviation. The second term in equation 4-8 is to correct for different magnitudes of standard deviation of different actions. Without this term there is a bias for actions with a higher variance, because they produce lower errors.

\[
\begin{bmatrix}
\epsilon_{nx} \\
\epsilon_{ny} \\
\epsilon_{nz}
\end{bmatrix} = \left| V_{n}^{\text{new}} - \bar{V}_{n} \right| \tag{4-7}
\]

\[
\begin{bmatrix}
\epsilon_{x} \\
\epsilon_{y} \\
\epsilon_{z}
\end{bmatrix} = \sum_{n=i}^{N} \frac{\epsilon_{n}}{\sigma_{n}} = \left[ \sum_{n=i}^{N} \sigma_{n}^{-1} \right]^{-1} \tag{4-8}
\]

\[
\epsilon = \epsilon_{x} + \epsilon_{y} + \epsilon_{z} \tag{4-9}
\]

Distance \( \epsilon \) is expected to be low when a recurring action is compared to the correct action in the database. Figure 4-2 shows an example of this effect for five different actions. Distance \( \epsilon \) can be used in two ways for recognition.

1. One way is to set a threshold for the maximum distance that is allowed from an action in the database. The value of this maximum distance should be higher than the expected value a new trajectory with this distribution. If the distance is lower than this threshold then the demonstration is recognized as being this action. The advantage of this method is that it can classify a demonstration as 'not in the database' when it is not matched to any of the actions in the database. This way a new never seen actions can be discovered.

2. The other way is to compare the recurring demonstration to every action in the database and recognize it as the action with the lowest distance. This method is more robust and can recognize actions even when the demonstration is a bit different from the database.

In chapter 6 the only the second method is used to test recognition performance for recurring actions.
Figure 4-2: An example of distance $\epsilon$ for five recurring actions each compared with five actions in the database. The labels on the x-axis indicate which recurring action (first letter) is compared to which action in the database (second letter). The letter codes are according to chapter 6. The blue boxes and red outliers in the boxplot indicate the range $\epsilon$ for 100 different comparisons. It can be seen that the error is much lower when the action in the database matches the recurring action.
Once an action is demonstrated and analyzed a robot should also be able to synthesize it. Synthesis is more than a replication of the observed motions. The method should be able to produce an action with never before seen initial conditions.

One of the advantages of the object-based approach of LfD is that the actions in the database are not coupled to any kinematic configuration such as a robotic arm. The trajectories that are generated, by the method described in this chapter, are thus also independent of a kinematic configuration. Only when these trajectories are tracked by a robot there is need to do inverse kinematics.

When synthesizing a trajectory knowing the allowed variances gives the freedom to divert from the optimal path when this is necessary for another goal. This can be used for:

- Obstacle avoidance
- Avoidance of kinematic singularities.
- Close to optimal synthesis on limited degrees of freedom robots
- Energy saving

In this chapter I will first explain how action trajectories with arbitrary start and end positions can be generated. In the second part of this chapter I will show how this trajectory can be tracked by a robotic arm. Finally I will present some suggestions to extend the possibilities of this method.

### 5-1 Trajectory synthesis

To synthesize an action trajectory four properties of the action are required:
The average trajectories $\bar{P}_n$ and $\bar{O}_n$ in the optimal reference frame. These are generated by equations 4-1 and 4-2.

- The average duration of the original trajectories $\bar{t}_{\text{end}}$ given by equation 3-9.

- The reference frame switching properties $\tau_1, \tau_2$ and $d\tau$ that were found by the optimization in section 3-4.

- A start and end position. The (relative) start position for the new situation can be provided by the same 3d object tracking system as was used for observing the demonstrations. The end position is part of a higher level planning (activity) that should be provided by a planning algorithm, or by the user.

Synthesizing a trajectory from this information is doing the inverse of a reference frame transformation as described in chapter 3-4. Reference frames pointing to the new start and end location are used to make a trajectory specific for this situation. Rotation matrices $R_n$ and scale $s_n$ to do the transformation are formed with equations 3-24 until 3-31. Properties $\tau_1, \tau_2$ and $d\tau$ are taken from the action database. Then the specific trajectory for this situation is obtained:

$$P_n = R_n \ast \frac{\bar{P}_n}{s_n} \quad (5-1)$$

$$O_n = Q(R_n) \otimes (\bar{O}_n) \quad (5-2)$$

This trajectory is still in normalized so the timing of each sample is given by:

$$t_n = \frac{n}{N} \ast \bar{t}_{\text{end}} \quad (5-3)$$

This is the relative trajectory that the operated object needs to follow with respect to the passive object in order to perform the action with the given start and end positions.

## 5-2 Inverse kinematics

In order to make a robot perform this action the trajectory needs to be transformed into robot specific joint instructions. This process is called inverse kinematics. Although it is not a fundamental part of the research in this thesis this section is included to show the method that was used to follow synthesized trajectories with a robot.

In order to do inverse kinematics one needs a forward-kinematic model of the robot. This model should provide the position and orientation of the end-effector $X$ based on joint angles $\theta$.

$$X = F_{\text{kin}}(\theta) \quad (5-4)$$

First initial joint configuration needs to be found that gives the desired initial position $X_{\text{des}}$.  

C. G. Rozemuller Master of Science Thesis
\[ X_{\text{des}} = \begin{pmatrix} P_{t_1} \\ O_{t_1} \end{pmatrix} \] (5-5)

I used a trust-region-reflective optimization \([44]\) to minimize the Euclidean distance between the desired position and the forward kinematic function.

\[
\min_{\theta} \| F_{\text{kin}}(\theta) - X_{\text{des}} \| \tag{5-6}
\]

This would provide one of the joint configurations that puts the end effector in the desired position. This solution might not be unique nor is it certain that it exists. In this thesis it is assumed that the robot is always able to reach this position.

Once an initial joint configuration has been found its Jacobian \(J(\theta)\) can be calculated numerically using the forward kinematics. With this Jacobian the amount of joint change \(\Delta \theta\) that is needed to go to the next desired state \(X_{\text{des}} = [P_{t_2}; O_{t_2}]\) can be approximated.

\[
\Delta \theta = J^{-1}(X_{\text{desired}} - X_{\text{current}}) \tag{5-7}
\]

Iteratively applying this step to every consecutive desired configuration in the trajectory results in the joint configurations that follow the trajectory. This simple method is however very sensitive for singularities and may thus result in unstable behavior of the arm. To avoid singularities I have used a more robust 'Damped least squares' method \([40]\) to inverse the Jacobian.

\[
\Delta \theta = J^t(JJ^t + \lambda^2 I)^{-1}(X_{\text{desired}} - X_{\text{current}}) \tag{5-8}
\]

In this equation \(\lambda\) is a damping constant that prevents the system from reaching singularities. As a byproduct of not being close to singularities it also prevents high joint accelerations. This property was used to tune the value of lambda to 0.2 where the maximum accelerations in the joints where near the limit of what the robot arm could follow. The downside of this method is that it is not able to follow the desired trajectory exactly. The size of the tracking error depends on the presence of singularities and the size of lambda. In the case of multiple start configurations each of these configurations will have different amounts of error. To find the optimal solution, in case there is more than one, a genetic algorithm was implemented that iterates the whole process in order to minimize the total error for the desired trajectory.

Using this method synthesized actions can be performed on a robot arm.

5-3 Advanced synthesis

The LfD method proposed in this thesis was designed to be suitable also for more advanced methods of synthesis. It was not in the scope of this thesis to work these methods out, but some recommendations for future work can be given based on this design choice.

The action database includes a measure of allowed variance in position and orientation. Earlier work by Ye \([10]\) or by Schneider \([9]\) has shown that such a statistical property can very well

Master of Science Thesis

C. G. Rozemuller
be used as an weight factor for the accuracy of trajectory following. This weight becomes relevant when multiple objectives need to be satisfied:

- It can be used to avoid kinematic singularities more effective when it is combined with the damping factor $\lambda$ that is now static. Then the trajectory would steer away from singularities when it is possible so it does not crash into them when the variance is supposed to be lower.

- It can be used to avoid other obstacles without affecting the critical parts of the action. This can be accomplished by making the desired position depend both on the optimal trajectory and the distance to other objects.

- It has been shown to be useful as a measure of stiffness for the robotic arm.

- It could be used to save energy, or minimize wear on the robot by avoiding situations where this occurs.

The alignment method can also provide a similar variance for timing. This would tell you when it is possible to slow down or speed up and when to be accurate. Then the objectives above can be achieved with even more flexibility in the action trajectory.
One of the important features of this new method is its ability to distinguish arbitrary movements from the essential movements. Therefore an action recognition experiment is used to test this feature. The experiment was designed to contain very similar actions with a high level of arbitrary features, so this feature is tested. The recognition is tested under multiple circumstances:

- Increasing the number of demonstrations in the database
- Having different persons perform the actions
- Disabling alignment
- Disabling reference frame optimization

Secondly also the real life performance of action synthesis on a robot is tested.

### 6-1 Pilot recognition experiment

A pilot experiment was conducted to find a suitable size of the action database for the recognition experiment. Four actions were demonstrated over ten times by one person to train the system. When trained with five demonstrations per action the system was able to identify recurring actions with over 90 percent accuracy. Demonstrating this many actions takes more than 45 minutes of the subjects time.

Based on this observation the conditions for the final experiment are ten demonstrations of five different actions. From this collection of demonstrations there are enough combinations of a training set and test set possible to get clear results. Because of time limitations and subject availability six different subjects performed in the final experiment.
6-2 Experiment set-up

For the recognition experiment six subjects have performed five different actions repeated ten times. This is a total of 300 demonstrated actions. This data is used to test the proposed method under multiple conditions.

The actions used in the experiment are tabletop actions that are commonly performed in household situations. Table 6-1 gives a description of each action and figure 6-1 shows a subject performing the actions.

<table>
<thead>
<tr>
<th>Label</th>
<th>Action</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Stacking</td>
<td>Placing one object on top of another object and removing it again.</td>
</tr>
<tr>
<td>B</td>
<td>Container</td>
<td>Placing one object into a narrow container and removing it again.</td>
</tr>
<tr>
<td>C</td>
<td>Sauce</td>
<td>Dispensing sauce into a bowl with a circular motion.</td>
</tr>
<tr>
<td>D</td>
<td>Shaking</td>
<td>Shaking the contents of a package into a bowl.</td>
</tr>
<tr>
<td>E</td>
<td>Pouring</td>
<td>Pouring the contents of a bottle into a cup.</td>
</tr>
</tbody>
</table>

In practice the first two and last two of these actions are very similar. This was done on purpose to challenge the method and find its limits. The subjects were instructed to pick random start and end object locations, however the action itself should be performed similarly. For example pouring was always done from on the right side of the cup, but the bottle could be placed anywhere.

The object tracking algorithm of the Point Cloud Library [15] was used to track the objects in the 3d data provided by the kinect 3d-camera. The algorithm uses a particle filter to find the object locations and orientations in every frame. The objects are selected by the user in the first frame. The use of this tracking system limits the objects that could be detected: The objects had to be large, non-reflective and have distinctive color features. Figure 6-1(f) shows an example of the object tracker output.
Figure 6-1: The five actions that are performed in the experiment. These images are stills from movies that were recorded during every from the demonstration.

Figure 6-1(f) is a Screenshot from the object tracker during the pouring action. In this screenshot is taken at the same time as image 6-1(e). The two tracked objects are highlighted in purple and blue. The orange and green dots represent different configurations that have been tested in this frame. The two boxes indicate the resulting object location and orientations. Most the background is filtered out to increase performance.
Table 6-2: A confusion matrix for subject 1 with eight actions in the database with 100 repetitions of each action.

<table>
<thead>
<tr>
<th></th>
<th>Stacking</th>
<th>Container</th>
<th>Sauce</th>
<th>Shaking</th>
<th>pouring</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stacking</td>
<td>100</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Container</td>
<td>4</td>
<td>96</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Sauce</td>
<td>10</td>
<td>2</td>
<td>88</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Shaking</td>
<td>6</td>
<td>4</td>
<td>4</td>
<td>86</td>
<td>0</td>
</tr>
<tr>
<td>pouring</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>100</td>
</tr>
</tbody>
</table>

6-3 Action recognition experiment

In the recognition experiment it is the performance of the new LfD method is tested. Also the added effects of alignment and RFO are evaluated. The alignment is expected to make recognition better because it removes variances that make the action less distinctive. RFO is required for the synthesis of actions, but should not decrease recognition performance.

Two realistic scenarios are used to test the recognition and synthesis: A personal situation and a general situation.

6-3-1 Personal situation

In this scenario the system has to recognize actions that have been demonstrated by a single person, and are for the test performed by that same person. This is the situation for a personal household robot that has been trained by its user. The results in figures 6-2(a) and 6-2(b) show the recognition rates for six different subjects with an increasing number of demonstrations per action in the database. For each person hundred different permutations of a training set and test demonstration were evaluated.

One can see that an average recognition rate of over 90 percent is achievable with three demonstrations in the database. This recognition rate is much higher than what can be expected from random choice, so the new action representation is functional. The recognition rate appears to stabilize at 95 percent when more than six demonstrations are used for training.

To identify what causes less than perfect performance a confusion matrix for subject 1 is shown in table 6-2. There it can be seen that the errors are caused because actions Stacking, Container and Sauce are mistaken for actions Container, Sauce and Shaking.

The classification of an recurring action is based on the value for $\epsilon$ that indicates the distance from actions in the database (as explained in chapter 4). There are two ways this classification can go wrong:

1. The value of $\epsilon$ for a correct comparison is still so high that it cannot be distinguished from values for incorrect comparisons. This is caused because the training set is not generic enough to cover the recurring but slightly different action. However making the training set more generic also increases the risk for the second error.

C. G. Rozemuller

Master of Science Thesis
Figure 6-2: Recognition rate in a personal situation with increasing training set size and with (a) different subjects, (b) different actions and (c) with alignment 'DTW' and/or Reference Frame Optimization 'RFO' disabled.
Figure 6-3: boxplot of distance $\epsilon$ when comparing five recurring actions (first letter) to the five actions in the database (second letter) for subject 1. The scores for correct comparisons (AA, BB, CC, DD, EE) should be lower than the other scores for correct recognition. In this case this is only reached by a small margin, causing incorrect classification for outliers.

2. The value of $\epsilon$ for an incorrect comparison is so low that it is mistaken for the correct classification. This is the case when two actions are very similar to each other.

Further investigation of the values of $\epsilon$ for subject 1 in figure 6-3 shows that both types of error have occurred. The scores for correct comparisons CC and DD are so high that they are not distinguishable from the others (error 1). And incorrect comparison BA gives scores that are so low that action B is mistaken for A (error 2). Since the experiment was designed to make actions A and B similar, this was as expected. An earlier example in figure 4-2 of subject 2, who has higher performance, shows that both errors are avoided by a much greater margin. Both errors can be avoided by setting a threshold for $\epsilon$ above which actions are marked as unknown.

Figure 6-2(c) shows the effect of disabling alignment and/or reference frame optimization. It can be seen that disabling these techniques does not have a big impact on the recognition rates. RFO was expected not to have influence on the recognition rate, however alignment was expected to improve performance. Inspection of the data (see appendix B) before and after applying the methods show that they operate as expected. This means that the variations they correct for are not of significant influence on the recognition scores. In this experiment the training and testing was done with the same person. Therefore it could be the case that the personal style of doing the action is so consequent that alignment is not required. Therefore a second recognition experiment is performed where personal style is not of influence.

With this result of 95 percent recognition rate it is shown that the method proposed in this thesis can be used for recognition of recurring actions in a personal situation.
6-3-2 General situation

The general situation is to test action recognition of subjects that are not in the database. This would be the case if you pre-install actions on a household robot, so you don’t have to perform the demonstrations yourself. In this experiment the demonstrations of up to five other subjects are randomly combined in a training set to recognize the actions of a sixth unknown subject.

Figure 6-4(a) shows a lower and much more scattered recognition rate for different actions in the general situation. This means that it is harder for the system to recognize actions that are not performed by the same person that trained it. Even combining thirty demonstrations by five subjects gives worse results than three demonstrations by the same subject. Again actions Sauce (C) and Shaking (D) have the worst recognition performance.

Lower recognition rates are a sign that there are more variations in the demonstrations of an action. This is the effect of combining the actions from multiple subjects, each with their own personal style. Therefore the action descriptions become more generic and less distinctive.

Figure 6-4(b) shows that enabling alignment now has a positive effect on the performance with recognition rates that are 5 percent higher. Aligning demonstration therefore reduces variations that are the cause of poor recognition performance.

There is no significant effect of RFO on the recognition rates. This means that reduction in variations that RFO accomplishes is not relevant to distinguish actions. RFO reduces variations in the start and end of the trajectory, so it could be the case that these parts of a demonstration are not specific for an action. RFO is however an integral part of the method required for synthesis so it should not be removed from this LfD method.
Figure 6-4: Recognition rate in a general situation with increasing training set size and with (a) different actions and (b) with alignment ‘DTW’ and/or Reference Frame Optimization ‘RFO’ disabled
6-4 Synthesis experiment

The ultimate goal of LfD is to effectively perform learned actions with a robot. In this experiment the five actions that have been learned are synthesized on an Universal Robots UR5 robotic arm (figure 6-5(a)). This robotic arm has 6 degrees of freedom and a reachable space that is large enough to perform the actions. The joint trajectories for the arm are synthesized with the method described in chapter 5 and tracked using a PD-controller. Grasping of objects is done using a Lacquey underactuated hand, shown in figure 6-5(b). This type of hand has six degrees of freedom but only a single actuator. The force of the actuator is therefore spread evenly among the contact points ensuring a good grip for any object shape [45].

The synthesis experiment was performed with two different datasets. The first ‘Personal situation’ dataset contains five demonstrations performed by subject 2. Subject 2 had the highest performance in the the recognition experiment. It is expected that this is also an indication for a good synthesis performance because this action representation contains many distinctive trajectory details. The second ‘General situation’ dataset contains thirty demonstrations from five subjects combined. This dataset had a lower performance in the recognition experiment and is therefore expected to be less successful in a synthesis experiment.

The novel RFO method was designed to makes actions independent of object location. To test this method every action is synthesized with two different object locations that have not been demonstrated. At location 1 the manipulated object is placed with an offset of [0.15, 0.15, 0] meter from the passive object. At location 2 this offset is [0.2, 0, 0] meter.

Table 6-3 shows the results of the synthesis experiment. The actions that are marked as ‘success’ have effectively been performed by the robot. This means that the functional goal of the action (as described in table 6-1) has been reached. Figure 6-6 shows images from the
Table 6-3: Results of the synthesis experiment for a personal and general situation. The actions were synthesized to start and end from two different locations. An experiment was marked as success when the action was effectively performed by the robot.

* There was not enough inclination and shaking to get the contents of the package in the bowl
** Water was poured from the bottle, but not enough to fill the cup.

<table>
<thead>
<tr>
<th></th>
<th>Personal situation</th>
<th>General situation</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>location 1</td>
<td>location 2</td>
</tr>
<tr>
<td>Stacking</td>
<td>success</td>
<td>success</td>
</tr>
<tr>
<td>Container</td>
<td>success</td>
<td>success</td>
</tr>
<tr>
<td>Sauce</td>
<td>success</td>
<td>success</td>
</tr>
<tr>
<td>Shaking</td>
<td>fail*</td>
<td>fail*</td>
</tr>
<tr>
<td>Pouring</td>
<td>success</td>
<td>success</td>
</tr>
</tbody>
</table>

robot arm performing every action.

In the personal situation four out of the five actions have successfully been synthesized. Only the shaking action did not synthesize effectively. It is notable that the pouring action was performed with real water without spilling a drop. In the general situation three actions were synthesized effectively. The actions that failed were pouring and shaking. It can be seen that synthesis from both locations performed equally well which means that the generalization with the novel RFO method worked.

During the experiment it could be seen that the shaking and pouring actions failed because the objects were not inclined enough to release their contents. Two causes for this phenomenon can be identified: First There is loss of trajectory details during the generalization. Although the demonstrations are aligned there is still so much variance in the trajectories that these crucial details are smoothed out when taking the average trajectory. Secondly the robotic arm is limited by its maximum acceleration, this means that a fast movement such as shaking cannot be tracked accurately.

This experiment has shown that four out of five actions learned from demonstrations can be synthesized on a robot when using the proposed method.
Figure 6-6: The actions as synthesized on the robot arm
The question that is answered in this thesis is: How can a robot learn a household task from human demonstrations in an unknown household environment, without the drawbacks of current methods?

A new LfD method was proposed in chapter 2 that combines current state of the art methods with novel innovations. With a recognition experiment it was tested if this method was able to learn and recognize household actions, without having the drawbacks of the current methods.

7-1 Conclusion

The proposed method uses an object-space statistical task description to represent actions. The representation can be made from human demonstrations that are recorded using a 3d-camera. As opposed to existing methods, this allows the subjects to perform the demonstrations naturally and in any environment. No models of the environment are needed on beforehand, there is no physical interaction with the robot and the room does not have to be equipped with special sensors.

In addition two novel methods are introduced to make recognition and synthesis robust to variations in demonstrations:

1. A new alignment algorithm can align multiple demonstrations with computing time that is only quadratic to the number of demonstrations. This is used to quickly make the task description independent of differences in timing from different demonstrations.

2. Reference Frame Optimization makes the trajectory independent of the start and end location of the objects. During synthesis this can be used to perform an action with never before seen object locations.

The experiments show that the proposed method can recognize actions with an accuracy of 95 percent. This is in case the system is trained with five or more demonstrations of the
same person. An average recognition rate of 80 to 85 percent can be reached when observing demonstrations of unknown subjects. This requires more than 10 training demonstrations by five different subjects.

The novel action alignment method contributes to a 5 percent increase in recognition performance, by reducing inter-subject timing differences.

The demonstrations can take place in a natural environment, without any special training or physical interaction with the robot. This is an improvement compared to current methods that require a specially equipped environment or physical guidance of the robot.

Results show that action synthesis on a robot arm is possible using the proposed LfD method. Four out of the five actions have successfully been synthesized. Using a novel ‘Reference Frame Optimization’ the trajectories could be generated for object configurations that have never been demonstrated.

Based on these observations it can be concluded that the proposed system can learn a household task from human demonstrations in an unknown household environment without having the drawbacks of the current methods.

7-2 Future work

Based on my experience with this method I have some recommendations for future work

One of the most limiting factors of this method is the kinect camera in combination with point cloud object tracking. The sensitivity of this system requires the use of large, distinctive objects. Especially with symmetric objects, or out of plane rotations the tracker is inaccurate and might drift. Also observing the demonstrations in bright sunlight does not work because of the camera. Improved 3d object tracking will directly affect the performance of this method.

The synthesis method can be extended to have more functionality, as was already mentioned in section 5-3. Synthesized actions should not only based on the average trajectory, but also based on the allowed variation in trajectory. When this feature is added the trajectory can be optimized to:

- Avoid kinematic singularities more effective when it is combined with the damping factor $\lambda$ that is now static. Then the trajectory would steer away from singularities when it is possible so it doesn’t crash into them when the variance is supposed to be lower.

- Avoid other obstacles without affecting the critical parts of the action. This can be accomplished by making the desired position depend both on the optimal trajectory and the position of other objects.

- Save energy, or minimize wear on the robot by avoiding situations where this occurs.

- And it has been shown to be useful as a measure of stiffness for the robotic arm.

The novel alignment method can also be used to measure a ‘time variance’ of the trajectories. This is a measure of when and how much the trajectories may change the timing. It is very similar as the variance in position that was used during this thesis. This property can be
used to change the timing of the trajectory for the same reasons as above, and for improved recognition.

The way demonstrations are performed by the users can be improved. With a guideline for the users it can be made sure that many of the variations that are now cause of poor performance are eliminated. Also the user can give spoken feedback to the system or use shared attention.

Incorporating object/affordance recognition makes this method applicable in a wider range of situations where the user cannot indicate the correct objects.

In this method the actions are described without any feedback. This limits the actions that can be learned. Humans use all sorts of feedback to correct their actions. Incorporating force, visual, audio, etc. feedback will therefore very much improve this LfD method. For example: when pouring a drink the robot will not stop when a smaller cup is already full.

With the method proposed in this thesis a robot was able to learn actions from human demonstrations in a situation that is very relevant for household robotics. With that result I hope to have contributed to the development of robots that can help people that are in need of constant assistance.
Appendix A

Learning from demonstration in literature

This chapter gives an overview of the current state of LfD in literature. Section A-1 summarizes the concerns from two surveys of the LfD research field. The research groups presented in section A-2 have proposed complete LfD frameworks. Most of the other research groups focused on more specific parts of LfD. Based on the subdivision made in the introduction all research is split up in three sections. Section A-3 focuses on how demonstrations are captured, how they are stored and what information they contain. Section A-4 shows how the captured demonstrations are generalized for use by a robot and section A-5 is about how the learned task can be executed by the robot.

A-1 Surveys

Weng et al. [46] and many others recognize a problem with the current state of the robotics field: Robotics have to make huge advances to be truly useful in a human environment. Current methods to create intelligent machines are: knowledge based (programming by experts), learning based (information feeding in a task specific learning program) and genetic search (random exploration of possibilities). However Weng states that none of these current techniques are powerful enough to match the capabilities of a human brain. The requirements for an autonomous learning system can be summarized in five points:

1. The system is not specific to tasks.
2. The tasks are unknown to the system designers.
3. The system can generate approaches for unknown tasks
4. The system has an online learning ability.
5. The system has an open-ended learning ability.
This article was written in 2001 and the field has advanced a lot since then, however this list of requirements is still relevant and should be considered for any general robot that needs to work in human environments.

Argall et al.[3] published a survey that summarizes the 2009 field of robot learning from demonstration. Based on this they addressed a few promising areas of LfD research that deserve more attention:

- The current state feature vectors describing the known world are too limited for use on a global scale.
- There is need to add temporal data to introduce a memory of relevant past events.
- Multi-robot demonstrations
- Evaluation metrics
- Execution failure awareness
- New techniques from learning from experience

These two surveys both emphasize the importance of thinking bigger than a single task. Especially the use of limiting state vectors is a point of attention that is relevant for this part of LfD research. I will focus this research on developing a framework that can learn any task that involves complex movements and varying objects of the same affordance. This can be part of a LfD framework that satisfies all conditions mentioned by Weng and Argall.

A-2 Research groups

The research groups presented in this section have all contributed complete LfD frameworks.

Hasegawa and his team developed a Self-Organizing incremental neural network (SOINN) [20, 4, 21, 22] that is capable of robust online pattern recognition with very basic prior knowledge. They developed a framework in which this technique is exploited to meet all properties that are required for a developmental system, as proposed by Weng [46] in the previous section. Especially the third point (The system can generate approaches for unknown tasks) is a problem to which not many solutions have been found. Hasagawa proposes a three layer-architecture consisting of an Input Layer, a Pattern Memory Layer and a Symbol Memory Layer, as can be seen in figure A-1. These layers respectively represent vectorized sensor output, recognized objects features and trajectories, and a world model with knowledge of preconditions and consequences. Using this they have been able to teach a robot spoken language without prior knowledge or a lexicon [20]. Simple sentences such as 'move the lemon around the melon' could be understood and executed. They also used SOINN for a maze solving robot that could navigate through recognized scenes, after having wandered around.

In a later article this framework was presented with another example: a robot that learned to ring a bell and wave in order to get food [21]. In 2011 they presented [22] a refined image
recognizer that can also categorize the images. The main novelty of this research group is the use of their robust online pattern recognition to analyze the scene for known and unknown objects and features. SOINN is one of the few existing techniques that can be considered for automatic recognition and categorization of new objects and can therefore be a valuable addition to a generalizing LfD framework.

Rüdiger Dillmann and Holger Friedrich [26, 27, 17, 47] were the first ones to propose a LfD framework that requires more user interaction than just a demonstration, as can be seen in figure A-2. This triggered a line of LfD research that other than demonstrations also inquires spoken instructions, advice, intentions, descriptions or suggestions. These techniques have a lot of advantages for a LfD system, since they can speed up and refine the learning process in a natural way. There is a wide variety of these technique, therefore we discuss them on an individual basis in section A-3-3. Later Rüdger Dillmann Worked with Jäkel [7, 39, 8] on automatic constraint recognition from multiple demonstrations.

Steil et al. [6] also pursues a multi-modal communication - including speech, vision, gestures and posture - to teach a robot. Figure A-3 shows the framework used for combining speech, vision and physical interaction. In this article they also propose the three timescales presented in section 1-4-2. They focus of their research on the fastest level of learning: situated learning. Situated learning should be a fast process so they consider imitation learning as an important candidate technique for this. They explain an architecture for situated learning that involves observation, internal simulation and sensorimotor exploration. This architecture is applied to a dexterous grasping task. A robot learns from observation which of its four basic grips is best for each type of object. They also propose that the field of datamining can significantly contribute to more sophisticated learning capabilities on the Ontogenetic time scale.

The loop of observation, internal simulation and sensorimotor exploration is a very good way to overcome the correspondence problem of mapping human demonstrations to robotic
actions. However simulations are costly and require a lot of knowledge about the environment which in our situation we don’t have. Learning by imitation seems therefore the other best way to handle learning on the quickest time scale.

Two of the most practical LfD robots that exist today come from Schneider [9] and from Ye [10]. They have independently of each other approached the LfD problem in a similar way. They capture the movements of a demonstration with great accuracy through the robot’s interface and apply a statistical analysis of the trajectory (further explained in section A-3-2). Both teams can show examples of task executions that approach a practical use. This makes them a good starting point for research on capturing trajectories. One drawback is the direct interface through the robot which we want to avoid in our framework.

A-3 Capturing the demonstration

A-3-1 Observation of objects

To learn a task the system must know to which objects the task can be applied. A conventional object recognizer is not enough. It must also recognize objects that have the same affordances. for example: A cup, a mug and a glass all have the ability to be poured in.

Hasagawas Self-Organizing incremental neural network (SOINN) network has the potential to this. SOINN was presented as a solution to many aspects of LfD. Hasegawa’s first LfD framework [20, 21] could learn language from demonstrations. It requires multiple SOINNs to be defined before showing the demonstration. Each concept used in the demonstration such as color, shape or complete objects needs a separate network. Each network requires a specific input vector containing just information about one concept, such as RGB values for color, shape primitives for shape or a combination of separate objects. With each demonstration also a spoken description is given. This way each object, word or task that is shown during the demonstration is associated to the part of the networks that represents its features. After this training the robot can recognize for example ‘green’, ‘3 objects’ and ‘object1
Figure A-3: multi-modal communication framework for LfD as proposed by Steil in [6]

moving around object2’. Trajectories are stored separately to be executed when needed. This information forms the basis for their speech understanding robot. Although this method has very impressive results, it does not take into account affordances of objects.

In 2011 Hasegawa [22] presented a more refined object recognition and classification method also using SOINN combined with Support Vector Machines. It was trained to recognize animals and their features (tail, furry) based on a small number of training pictures. After being initialized with this small number of training images it could incrementally improve itself learning from newly presented cases, without offline processing. This method shows the functionality that is needed for affordance recognition.

SOINN has many variants and has been applied to many other applications that require image recognition, including maze solving [4] and Internet-based object recognition.

SOINN can be used for classification of known and unknown objects which is needed for our framework. However, to do so it either needs suitable feature vectors or a set of supervised examples to start with. In our case of pouring it would need feature vectors that can discriminate between a cup and a bottle. Providing a set of examples is also not a practical technique for a household robot, because it takes too much effort.

Many alternative affordance classification methods can be found in the field of image recognition. Kjellström [23] proposes to classify objects not only by visual appearance, but to also incorporate typical movements of an object. For example: if it looks and moves like a hammer, then it is a hammer. They intend to learn object affordance from observations of humans performing household tasks. An FCRF is trained both with object and action features, and the likelihood of each combination.

Combining object appearance with typical movements is a feasible option in LfD since both data are available per definition. Especially when a lot of data is available this is advantageous. For example when the robot is still observing while not learning or executing a...
task. Motions can be well combined with the SOINN as an extra feature vector for object recognition. But this method is not suitable for fast learning.

Two methods that can work on a faster time scale are given by Madry and by Grabner. Madry [24] searches for visual features in objects that can be associated with grabbing techniques that allow special actions (pouring, eating, playing). Grabner [25] uses a 3d map of the environment to find suitable sitting places. In a simulation he applies sitting motions to every position in the map to check which objects have chair-like affordances.

Both methods check the affordance of an object by (virtually) applying actions. This seems like a good starting point for quick affordance recognition in LfD.

A completely different method is found outside the classical domain of image recognition. D’Este uses constraint generalization over multiple demonstrations to get a definition of objects and concepts. The first time an object or concept is seen then every feature describing it is stored as a rule. Each time it is seen again in a different form (told by a supervisor) the rules are generalized to incorporate all encounters. For example: After a few supervised encounters a pillar was described by the system as: ‘above(A, B), below(B, A), yellow(A), medium(A), squat(A), column(B)’, where A and B represent two arbitrary objects.

This is a very interesting method, but it requires constant supervision and a lot of examples. Just like some of the previous methods this is feasible to use in a ‘life long learning’ system.

A-3-2 Observation of motions

Capturing and synthesizing motions is a crucial aspect for a household robot LfD framework. In many cases it is a special motion that defines a household task. For example the cleaning of a plate, folding the laundry or pouring a glass. These motions are so diverse and object specific to be preprogrammed so they need to be learned.

Krüger et al. proposed a method [16] similar to phonemes and grammar in speech to describe human actions. There is evidence ([48] among others) that just like phonemes in speech there is a limited number of prime movements from which all actions can be built. This segmentation is applied both to the motion of objects and human bodies. PHMM (Parameterized Hidden Markov Models) were applied to describe the motion independent of the direction. The similar motions of multiple demonstrations are segmented into a sequence of Gaussian distributions. By grouping the longest motion sequences that are common in several demonstrations together, they identified action primitives. Using this they can synthesize and recognize actions. They could imitate a ‘peg in the hole’ task after demonstration and they were able to quickly predict the position on the table that a person was trying to reach.

PHMMs are a very suitable way of describing motions. It both supports segmentation of motions as well as the use of variance analysis (A technique described in the next paragraph). The requirement for multiple demonstrations is a downside of the use of any statistical technique to describe motion.
Not every action during a demonstration is important. For example, when pouring a drink it is not important what path you use to pick a cup, but the pouring motion is very specific. Several research groups have found methods to differentiate the important actions from the arbitrary actions. Schneider et al. [9] taught a robot to make coffee, put an ice cube in a cup or to pour a drink. The robot first locates all objects on the scene. A 6-dof manipulator robot can learn a new task by having its manipulator physically guided by a human teacher. After multiple demonstrations it looks for the relative relation between object locations and manipulator path using a mathematical technique called 'local, heteroscedastic Gaussian process regression'. Movements with low variance in manipulator orientation with respect to an object are considered important. The sequence of these important movements with respect to the corresponding objects form the basis of the task.

Ye and Altarovitz [10] also used variance analysis as a basis for their motion planner. Similar motions are aligned using Dynamic Time Warping (DTW) so their variance can be calculated. The variance forms the basis for a cost function used by a path planner that is used to avoid obstacles.

Both of these articles use techniques that extract the important features from the demonstration using the variance of multiple motions. This could form the basis for a task description that is independent of relative object location, but can incorporate special movements such as pouring or pushing a button.

In 1999 Rudger Dillmann proposed another technique [17] that is now often used to find the key moments in a demonstration. A data glove captures the exact trajectory of each finger. Moments of low total velocity are considered to be the key poses in a demonstration. These different key poses can be recognized in multiple demonstrations. These key poses are recognized by a neural network to represent one of the robot's basic operations (for instance different types of grabs).

This technique has proven itself in later work and can be considered a valuable for segmentation. However, a very accurate hand posture recognition was used in these experiments. A data glove is not a solution for our system, therefore it should be checked to what extend velocity is a good indicator of key moments when using camera based system. This method has the advantage over variance based techniques that it only requires a single demonstration.

Steil et al. [6] also uses hand posture recognition to teach grabbing techniques to a robot. They compare two methods: The first 'naive' method transfers the hand posture from the demonstrator to the robot hand, with adaptation for the fact that the robot has only three fingers. This method was not successful. The second technique matches the demonstrated posture to one of the four initial conditions for known grabs. A reinforcement learner uses this as an initial condition in the search for a good grip. The result of this simulation is then executed on the robot hand, using tactile feedback to evaluate the stability. The loop of observation, simulation and evaluation can be repeated until a successful technique is found.

Steil has shown that copying the demonstration is not always a successful method. It makes sense to iterate through a number of learning techniques to learn successful strategies, however as discussed before simulation is not a feasible option in our case. It is clever to use tactile information as a performance measure.
A-3-3 Observation and user interaction

Learning from Demonstration aims to create a learning interface that is as natural as possible. It is in the nature of humans to interact with more means than just demonstrations. Valuable information about the demonstrated task can be acquired from the teacher when these possibilities are exploited. LfD researchers have come up with a wide variety of methods to interact with the demonstrator in order to improve learning.

Dillmann [26, 27] has shown how inquiring user intentions helps the system to generalize a pick and place task. The user is asked for the intentions of all actions to find out if they are intentional, forced by environment or unintentional. Also the user can adapt the preconditions that the system assumes after having processed the demonstration. User intentions become relevant when the framework is expanded to learn more complex tasks.

Some researchers use speech as the main input method, providing a clear structure for the demonstrations.

Nicolescu [28] uses natural spoken instruction to teach a mobile robot a specific pick and place task. During the learning phase the robot follows the teacher. When the instruction 'here' is given the robot adds the nearest object as a landmark. With the commands 'Take' and 'Drop' the robot can be instructed to manipulate objects. Then the robot executes the task and listens for feedback from the teacher. The teacher uses direct commands that program the robot to do a specific part different or skip it entirely.

Rybsky [29] showed a learning technique that uses a speech based user input, combined with location awareness to program a robot. This robot could by speech learn an if...then...else structure in which preprogrammed tasks could be executed. During the demonstration it would follow the instructor to learn the specific locations and persons involved with the tasks. It was also able to recognize missing statements in the structure and specifically ask for correction. The robot could also report its learned program back to the instructor for validation.

Fritz et al. [30] have constructed a framework in which natural language instructions can be refined with user demonstrations. Natural language tends to be unsuitable for programming because of the many omissions. With this framework the missing arguments and instructions can be obtained from demonstrations.

Spoken instructions are not suitable as main input for the type of tasks we intend to do. They lack specificity to constrain complicated movements such as pouring. Although it is possible to use speech to provide structure to a demonstration. For example by indicating the topic of the demonstration for future reference or by indicating which part of the demonstration was wrong and needs refinement.

Humans naturally tend to give specific advice when a demonstration goes wrong or needs refinement as was described by Thomaz et al. [32]. They investigated the human teaching behavior when interacting with robot learners. Non-expert people were asked to teach an RL algorithm to bake a virtual cake, only by giving feedback. Their finding was that humans naturally want to guide the agent with details, although it only responds to general feedback.
Teachers will adjust their behavior as they develop a model of the agent and they are biased to give more positive than negative feedback. Teachers wanted to see the effect of their negative feedback sooner, so an undo button improved their scores significantly. Two different types of positive feedback can be distinguished: motivational and instructive. With these findings incorporated the learning algorithm converged much faster.

This knowledge of teacher behavior is very valuable when feedback of any kind is incorporated in the framework.

Some LfD frameworks already exist that take advice from a teacher

Argall [33] proposes to use Advice-Operators that can quickly correct demonstrations. For example she uses advice formulated as ‘wider’ and ‘faster’ to mathematically alter the demonstrated policy of an inverted pendulum robot. In a later paper [34] she showed that weighting input from multiple teachers is beneficial. A driving policy was optimized by combining and weighting demonstrations from several teachers. The result was a policy that outperformed the teachers even if they were able to select their best performances. However it was still worse than advice-corrected teacher policies.

Maclin [35] created intelligent game agents that were trained by RL and external human advice. The human was able to observe the progress of the learning algorithm and could make suggestions of new policies using an advice programming language. With this advice taken into account as an extra nodes in a neural network, the agents survived longer. Combining two advices worked even better.

Kullman [36] used human advice to improve strategies in a robot soccer ‘keepway’ game. He translated spoken advice into semantic rules that are incorporated in a Neural network. A reinforcement learner is then used to improve the total strategy. Although the advice was simple and logical, but not thought of by an expert, there was an improvement in performance for every rule. Strangely the combining all advices did not result in a performance gain as big as by the single advices.

All of these methods of advising have great benefits for a robotic learner. However all these methods require a framework to begin with. Therefore they are not yet in the scope of this research. They should not be forgotten when this research is continued.

Shared attention is the form of user interaction that receives most attention in LfD research. Shared attention is the technical term that indicates that multiple individuals (or in our case, robot and teacher) have shared focus on the same object. With humans this is naturally accomplished by gazing and pointing.

It is used by Steil [6] to select the target object by recognizing a pointing hand posture. But to an even greater extend it was used by Hasegawa [20, 21] in his framework to target the subject of a spoken description.

Similiarly D’Este [5] learned a robot to perceive and recognize objects based on their features (color, shape, size, weight, position) and how they are combined: ‘object that is round and yellow is a ball’. The robot uses a dialogue system to refine its knowledge on feature concepts. An example dialogue is given in figure A-4.
Holzapfel [31] uses the same kind of dialogue to let a robot discover new objects and their intended use from spoken descriptions.

Shared attention is a great tool to discover and isolate new objects so their features can be learned by an object recognizer. This does however add an extra step to the learning method making it less quick and flexible. Another use of shared attention is to give targeted advice or correction. Both uses are currently not in the scope of this research but should be considered later.

Many other researches avoid the problems of capturing a demonstration by letting the demonstrator directly control the robot. Parts of the stored trajectories can then directly be used for imitation. Direct interaction with the robot is not desirable for our task.

A-4 Task generalization

Once the Demonstration is captured its description should be generalized so it is applicable to every variation of the task that may occur. As shown in section 1-4 there are many levels of generalization possible. Steil [6] describes them as different time scales and Krüger as different scales of actions (action primitives, actions and tasks). For this research we specifically look for methods that are robust against variations in:

- Location of involved objects
- Shape, looks and variant of objects
- Speed of execution
In literature three main techniques are used to generalize captured user data into a task description that is usable for the robot: symbolic logic, Statistics and Reinforcement Learning. (symbolic) logic searches for a set of rules that describes a task most generally. For example: 'In every demonstration the bottle only tilted when it was held above the cup, so this must be a rule'. A second group uses (Bayesian) statistics to find the most probable action for each situation: 'The bottle was tilted in 80% of the cases when it was held above the cup, so this is the most probable action'. The last group uses reinforcement learning to find the best action for a certain situation: 'after simulating thousands possibilities, tilting the bottle when above the cup was the most succesfull action'

All these methods generalize over multiple demonstrations, however some researchers use only one. For example by mapping each step in the demonstration to a known action by the robot. This has the advantage of only needing only a single demonstration, also called 'one shot learning'. The disadvantage is that this method cannot deviate from a known skill set. This is not a problem when it concerns simple pick and place tasks, but in our case we want the robot to learn concepts such as pouring which are not yet a skill of the robot. A more advanced method may also mimic the important parts of the trajectory. It could use speed as a weight factor of the importance of the trajectory (as done for hand postures by Dillmann [17]). In the rest of this chapter we will discuss methods that generalize over multiple demonstrations.

A-4-1 Logic

Most research groups in this category use pre- and post conditions to describe a task. They segment the task into smaller actions and assume that each action is done for a reason. This reason might be fulfilled by a previous action or by another event. Logic deductions can be used to find the rules of combining the right actions into a task.

Ekvall [37] uses a framework that can analyze pick and place tasks. In this research all objects were known on forehand or simulated. The skills of the robot are domain operations 'pickUp' and 'putDown'. To avoid problems it uses domain predicates that check for: the state of the hand, location availability, collisions, reachability and temporal constraints. With these predicates the pre- and postconditions for pickup (e.g. empty hand) and putdown are defined. This way each pick and place task can be analyzed and planned. Human advice (e.g. first move this box so there is place for another object) can be taken in account by adding preconditions, or these preconditions can automatically be generated from multiple examples by comparing the similarities in the state sequence.

Hasagawa’s maze solving robot [4] can recognize its starting and goal position, and it knows the relations of all positions in between. These transfer possibilities are stored as rules so logic can be used to find a path. A later framework [21] presents a solution for general problem solving. From each demonstration the cause-effect relations are stored in the form of pre-conditions, a deletion list and an addition list. The article does not specify how these relations can be found. The use of a deletion and addition is unique in the literature on LfD, but it is crucial to describe how objects can be discovered or removed.
Dillmann’s demonstrations [26, 27, 17] are captured and stored as a sequence of known actions such as approach, grip, and transfer. This sequence of basic operations is combined with knowledge from user intentions to generate branches in the demonstration sequence. If part of a branch is indicated to be unnecessary it is deleted or if it is conditional an if...then...else structure is applied. Parts of the sequence that handle one specific object are grouped together (object groups). Then the Object Selection Criteria for each object group are determined based on shape, relation (in, on) and user input, so it is known which object to apply this action on.

Konidaris [38] describes the CST (Constructing Skill Trees) algorithm that segments and combines the skills learned from multiple demonstrations to a general skill tree. To solve a maze they obtain control policies from multiple demonstrations that cover parts of the maze, and they use smart segmentation techniques to combine these policies.

Nicolescu [28] uses nodes in a graph to represent each instruction. With each demonstration the graph is expanded, the common sequences are recognized and the differences are treated as alternative options. Alternative options with a longer path are ignored and the robot will always choose the simplest alternative.

Kruger [16] uses a part of this technique to describe his action primitives as the lowest level of the action grammar. They are defined as the longest sequence of movements that are common in several demonstrations or at several times during the demonstration.

Looking for pre and post conditions is a very powerful method to understand complex tasks that have many different solutions. In our case we may assume that the task has only one symbolic solution (‘pick up bottle and pour it in the cup’) with variations that cannot be described symbolically such as different positions and shapes. Segmentation by longest common sequence (as used by Kruger and Nicolescu) can be useful in our case. It can be used to compress the task description.

Another approach assumes that every demonstration contains the same sequence of actions. They do not look for pre- and post conditions but for general constraints that describe all demonstrations.

Jäkel [7] applies automatic constraint generalization to learn manipulation sequences. This article shows the example of pouring a drink. All the objects used in the demonstration are modeled in detail and trajectories are tracked using MoCap and datagloves. Dillmann’s method [17] is used to segment demonstrated hand movements at key moments, so a sequence of detailed hand poses is obtained. All key poses form the nodes of a graph that represents the motion for each hand. Then a large number of possible constraints is generated for each node in the graph based on relative positions and movements of all frames involved. Also key regions for objects (for example a bottle opening should be in the area above the cup) are considered as constraints. The last step involves the teacher showing many different examples of the demonstration (assumed to have the same key moments) so only the most general constraints are left.

In a next research [39] they used a simulated environment to further generalize these constraints. Using a parallel evolutionary algorithm they iterated different combinations of constraints, until the configuration with the most constraints that still describes the general

---

1Of course this depends on the definition of symbolic
situation was found. Recently this work [8] was expanded with contact constraints and also
the frames of the fingertips and wrists are considered in the search for constraints. As an
example they used the case of putting a bottle in a refrigerator at any door angles.

This method allows to find suitable generalization for exactly the type of task desired for
this research, however it relies on very accurate measurements and pre-modeled objects. The
target set up does not have these resources, so this method is not suitable.

**A-4-2 Statistical**

Only a few researchers use classical statistical methods to generalize their task. Moldovan
[49] uses SRL (Statistical Relation Learning) to learn object affordances to a robot. This
algorithm uses statistics to capture the dependencies between actions, object properties and
effects from demonstration in a Problog language. The captured probabilities can later be
used to select the most likely action causing a desired effect on an object. These types of
methods can only generalize known effects for known actions, which makes it unsuitable for
our system since we want to learn new effects and actions.

The variance methods used by Schneider et al. [9] and Ye and Altarovitz [10] (described in
section A-3-2) are better to describe new phenomena.

Ekvall [37] uses a similar method to find relations within object locations in a ‘table setting’
setting. To find which objects are placed relative to each other he calculates the position
variances of all objects to each other or to a reference frame. The objects with the lowest
variances are most probable to have a relation. For example: the spoon is always placed
relative to the plate.

These methods provide you with the most general form of the task (the mean) and the
variance provides nice limits for changes in tasks execution.

**A-4-3 Reinforcement learning**

Reinforcement learning from demonstration takes a much different approach to LfD. These
researchers use the demonstrations as a starting point for their much broader search for so-
lutions.

The method of Steil [6] was already mentioned in section A-3-2. It uses Reinforcement learning
to find the best grabbing technique based on initial conditions from the demonstration.

Taylor [50] combined learning from demonstration (LfD) with Reinforcement learning (RL)
using transfer learning methods. A ‘keepway’ game was demonstrated by human teach-
ers of different performance. Using different transfer methods (Value Bonus, Extra Action,
Probabilistic Policy Reuse) this was used as a basis for a RL algorithm. This resulted in
an significant performance improvement even when the teachers were ‘handicapped’ or only
their worst performance was used.

Atkeson [51] used a human pendulum swingup as a basis for their RL algorithm.
Once a feasible task description was found these methods can be used to efficiently execute this task on a robot. All of these methods have one big disadvantage: they need a lot of iterations to find a solution. Doing these iterations with the robot takes a lot of time. Once an initial solution is given this process can train itself without supervision. Speeding up this process would require a simulation, but that requires knowledge about the environment which is usually not available. Research is being done on obtaining models for unknown objects and situations, but this is not yet usable for this research.

A-5 Task execution

Once a general description of the task is found most researchers directly apply the logical or most probable sequence of actions to the presented situation.

Hasegawa’s maze solving robot has learned the rules to move between different positions. When represented with a task it can analyze these rules for the optimal sequence of actions. His latest framework framework [21] can be presented with tasks such as 'Put an apple on the table when there is nothing on the table'. It uses a General Problem Solver to find the logical combination of actions on the objects based on the pre-conditions, addition list and deletion list.

This direct method works fine in most situations, especially under lab conditions.

Some researchers however adapt the solution to the newly presented situation in which unexpected (not presented in a demonstration) things may have happened. For example an object may block the path that is found for pouring. For these situations pre-programmed skills are needed.

In Jäkels work [7, 39, 8] all the constrained orientations of all key moments of a task are found, however the path in between is left open. The motions between these key moments are planned by a motion planner so other constraints such as joint loads and obstacle avoidance can be incorporated.

In the work of Ye [10] there is also a path planner that uses the variance in the demonstrations as a weight for deviations from the original path.

Elvall [37] incorporates the solutions of a path planner, collision avoider and grab analyzer as preconditions in his pick and place framework.

These examples show that there is still need for some pre-programmed skills to ensure robust execution of a task. A problem is that these pre-programmed skills may conflict with new skills that the demonstrator wants to teach. For example: a method that always uses a collision avoider can never learn to push a button. Good segmentation of the tasks and required skills are needed to avoid these situations.

Lastly there is a method that plans its task based on a task description as given by the user: In Hasegawa’s work on language understanding [20] there is no information stored about specific tasks. After understanding the concepts of a words such as 'green', 'moving around'
it analyzes each spoken task instruction for known words and uses them to select targets and apply a the known trajectory (the verb in the sentence) to these objects.

It makes sense to use all of the information given by the user in the task description. However this is outside the scope the current research
Appendix B

Data plots
Figure B-1: Raw relative trajectories
Figure B-2: Raw relative trajectories after trimming
Figure B-3: feature vectors
Figure B-4: aligned and normalized data
Figure B-5: aligned, normalized and reference frame optimized data


[40] S. R. Buss, “Introduction to inverse kinematics with jacobian transpose, pseudoinverse and damped least squares methods,”


