

# Integrating Object and Grasp Recognition for Dynamic Scene Interpretation

Staffan Ekvall and Danica Kragic

Computational Vision and Active Perception Laboratory

Centre for Autonomous Systems

Department of Numerical Analysis and Computer Science

Royal Institute of Technology, Stockholm, Sweden

Email: {ekvall, danik}@nada.kth.se

**Abstract**— Understanding and interpreting dynamic scenes and activities is a very challenging problem. In this paper we present a system capable of learning robot tasks from demonstration. Classical robot task programming requires an experienced programmer and a lot of tedious work. In contrast, Programming by Demonstration is a flexible framework that reduces the complexity of programming robot tasks, and allows end-users to demonstrate the tasks instead of writing code. We present our recent steps towards this goal. A system for learning pick-and-place tasks by manually demonstrating them is presented. Each demonstrated task is described by an abstract model involving a set of simple tasks such as what object is moved, where it is moved, and which grasp type was used to move it.

## I. INTRODUCTION

The next generation of robots will be placed in our homes and help us in our everyday lives. The number of possible tasks these robots may aid us in is huge, and we cannot expect the robots to arrive pre-programmed for all these tasks. Neither can we expect users to program tasks by writing code. Instead, we need a system that enables the user to demonstrate the task, and equip the robot with the ability to learn from observation. This paper describes our recent steps towards designing a programming by demonstration system.

The project goal is to create a system that allows a human to demonstrate a task, which is later executed by a robot. In general, the initial task setting changes between the demonstration time and execution time. Therefore, it is not sufficient to just replicate the human movements. Instead, the system must have intelligent modules that understand what has been demonstrated. We have initially constrained the system to pick-and-place tasks.

### A. Programming by Demonstration

The basic idea in the field of *Programming by Demonstration* is to develop a system able of learning tasks demonstrated by a human teacher [1]. Figure 1 describes our general PbD process, inspired by the process flow suggested by [2].

- **Demonstration**

The user demonstrates the task, and various sensors such as cameras observe and store the demonstration.

- **Action Segmentation**

Irrelevant sensor data is filtered out, and actions are identified and segmented.

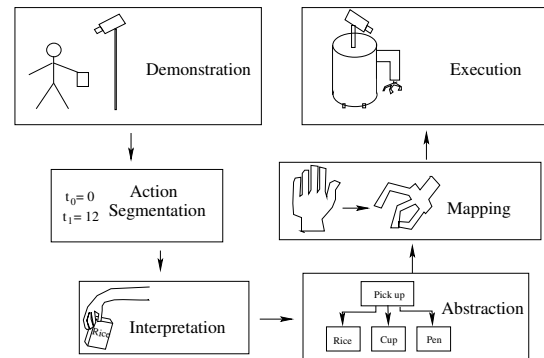


Fig. 1. The PbD process.

- **Interpretation**

The sensor data is interpreted into meaningful actions, such as *Pick up cup*.

- **Abstraction**

The interpretation of each subtask is stored and the entire demonstration is abstracted into a set of linked key identifiers.

- **Mapping**

From the abstraction symbols, grasp types and trajectories are calculated for the robotic platform. This mapping process may be based on demonstrated examples.

- **Execution**

The task is executed by the robot.

The abstraction design depends on the purpose of the system. In general, each task is decomposed into a number of actions, e.g., *Pick up*, *Put down*, *Insert* etc. Each action is linked to one or more objects. Each manipulative action requires a specific grasp, that depends on the object and the upcoming task. Hence, recognizing both the manipulated object and the grasp type is important in order to replicate the task. The process is illustrated in Figure 2.

This paper describes the first four PbD process steps in our system. The interpretation system is built from two independent modules: Object Recognition and Grasp Recognition. The objects class and location are estimated by the Object Recognition module, and the grasp used for manipulation is

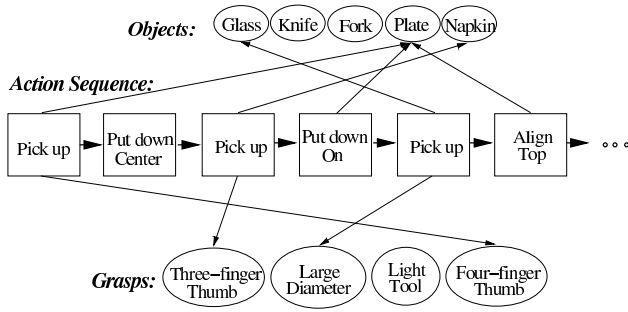


Fig. 2. The task set table, decomposed into several subtasks.

estimated by the Grasp Recognition module.

At this point, we concentrate on understanding *what* has been demonstrated, rather than *how* to execute the task. The latter problem is also very important and several ideas have been presented in the last few years. A promising approach is to map the sensor readings onto primitives [3], thereby reducing the dimensionality of the problem. The primitives constitute learning examples for learning an optimal perception-action mapping policy.

This paper is organized as follows. In Section II we review some of the related work in this field. We then describe each of the first three steps in the PbD process. We begin with Demonstration, in Section III. The technical details of the system are described in Section IV, Interpretation, and Section V, Action Segmentation. The system is evaluated in Section VI. Finally, Section VII concludes this paper.

## II. RELATED WORK

There have been numerous advances in PbD technology in the last few years. In [4], Dillmann *et al.* present a PbD system. Here, grasps are recognized using a neural network. Stereo cameras are used for hand pose estimation and object detection. They not only recognize the task, but also analyze the trajectory taken by the human to execute the task by mapping the trajectory onto a set of elementary operations (EO). An EO is a basic low-level movement, for example *linear-move*, *free-move* or *spline-move* etc. Then, these EOs are mapped to the robot platform at execution time.

In [5], Ogawara *et al.* describe a method for learning from multiple observations. They argue that a task-level approach, such as the one described in this paper, may very well recognize a number of task primitives. However, we cannot be sure which detected interactions are essential to complete the task, without prior task-dependent knowledge which should be obtained from observation. By observing multiple demonstrations, these ambiguities may be avoided. The system generalizes over all demonstrations, and irrelevant actions are thereby filtered out, a conclusion also presented by Nicolescu *et al.*, in [6].

Other PbD approaches rely on demonstrations in a virtual environment [1], [7]. In [8], such a system is presented. The user interacts with virtual objects using a CyberTouch glove and records the task. Then, the task may be executed in a real

environment by a PUMA 560 robot. Currently, the system allows to pick and place objects on a working plane, to stack objects, and to perform peg-in-hole tasks. The major advantage of a virtual environment is that sensor information is perfect, and there is no doubt what object has been moved, or where it has been moved. The main drawback is that the system is limited to the preprogrammed virtual environment, and the available objects that are modeled in 3D. Hence, it is not likely that the approach will be applicable in dynamic and complex environments, such as peoples' homes. In addition, it is certainly more natural to demonstrate a task in the real world than in a virtual one.

In [9], Kang and Ikeuchi describe techniques for solving the action segmentation problem. By analyzing the fingertip polygon area, and the speed of the hand, different phases of a grasp, the pregrasp, grasp and manipulation phase, are identified. The fingertip polygon area is an indication of the hand preshape while the speed of the hand is an indication of the hand transportation.

## III. DEMONSTRATION

The user demonstrates the task by manipulating objects in a natural way. The objects are arranged to the desired configuration, and the system then automatically builds the task description. The user notifies the system when the demonstration starts and when it is complete. Figure 3 shows the experimental setup used in this work.



Fig. 3. The PbD experimental setup.

### A. Measurement System

Two types of sensors are used in the demonstration phase. For object recognition, we have used an ordinary camera. The camera is mounted at a specific, known location which allows us to calculate world coordinates from screen coordinates. The camera could also be used for grasp recognition. In [10], Ogawara *et al.* demonstrates a grasp recognition system using an infrared camera. However, the hand occlusion problem during the grasping of an object remains. In our work we have used the Nest of Birds magnetic tracker for grasp recognition.

Nest of Birds consists of an electronics unit, a transmitter and four pose measuring sensors. The sensors measure transmitter-generated magnetic fields. The electronic unit controls the transmitted signals and directs the sensor measurement. From signals measured by the sensors, Nest of Birds

calculates the position and orientation of each sensor. Thus, each sensor has six degrees of freedom.

The sensors are mounted on a glove, illustrated in Figure 4. The center sensor, mounted on the back side of the glove, serves as a reference sensor. It measures the position and orientation of the hand. The remaining three sensors are mounted on the thumb, index finger and little finger, and provide position measurements of the fingertips.



Fig. 4. The glove used for human input.

#### IV. INTERPRETATION

The interpretation of user actions is performed by two separate modules. First, the grasp is recognized from the magnetic tracker data. Then, the vision system recognizes the object as it is being moved.

##### A. Grasp Recognition and Mapping

As mentioned, recognizing the grasp taken by the human user is an important step in the PbD process. For this purpose, we have used Hidden Markov Models (HMMs) to model the hand posture transitions during the grasp. The hand posture is captured by a magnetic tracker, which requires the user to wear a dataglove. The HMM approach is described in detail in [11].

In order to construct a suitable HMM for modeling, the number of states  $N$  has to be selected as well as the number of possible discrete observations  $L$ . In addition, the probability density matrices  $A$ ,  $B$ , and  $\pi$  have to be determined by training. The most common form of HMM is the so called 'left-to-right' HMM. Such a model always starts in the first state, and ends in the last state. It is useful because many real applications have the same nature.

Using a HMM allows us to model the entire grasp sequence, and not just the final posture. The input to the HMM are different finger combinations, *hand postures*. For each posture type, the HMM builds a statistical representation of the probabilities of observing different fingertip locations. The state of the HMM is the estimated current posture of the hand. When grasping an object, the user is assumed to start with the hand opened, move the hand to the object, and finally wrap his fingers around the object. Thus, it is natural to use a left-to-right HMM, as no grasp type involves closing and then opening the hand again.

1) *Feature Extraction*: Each of the sensors on the data glove estimates a 3D-position  $\mathbf{p}$ , calculated according to Equation 1, providing a total of nine values. The position of the sensor is represented by  $\mathbf{x} = (x, y, z)$ , and the reference sensor is represented by  $\mathbf{x}_r = (x_r, y_r, z_r)$ . The rotation matrix  $M$  is calculated from the Euler angles  $\phi$ ,  $\theta$  and  $\gamma$  according to Equation 2. As seen in Equation 1, the features derived from the sensors are both translational- and rotational-invariant, since the positions are multiplied by the transpose (inverse) of the rotation matrix.

$$\mathbf{p} = M^T (\mathbf{x} - \mathbf{x}_r) \quad (1)$$

$$M = \begin{pmatrix} c_\phi c_\theta & s_\phi c_\theta & -s_\theta \\ -s_\phi c_\gamma + c_\phi s_\theta s_\gamma & c_\phi c_\gamma + s_\phi s_\theta s_\gamma & c_\theta s_\gamma \\ s_\phi s_\gamma + c_\phi s_\theta c_\gamma & -c_\phi s_\gamma + s_\phi s_\theta c_\gamma & c_\theta c_\gamma \end{pmatrix} \quad (2)$$

$(c_\phi = \cos(\phi), s_\theta = \sin(\theta) \dots)$

2) *Grasp Modeling*: The three fingertip locations are quantized into one of  $L = 50$  cluster positions. Each cluster is 9-dimensional, since we have three fingertip locations with three dimensions each. The clusters are found using K-means clustering on the training data. The number of clusters should be low enough to gather enough statistical data of each posture, but high enough not to lose too much accuracy. We found 50 clusters to be a good compromise, although the result was about the same for 40-80 clusters.

We assign a left-to-right structure to our HMM, and each model operates in parallel. Each state in the HMM represents a hand posture, and a grasp sequence is in fact a sequence of hand postures. Therefore, HMMs are ideal to model grasp sequences. With 50 different hand postures, a grasp sequence is typically constructed of 5 postures. Thus, we have used 5 states in each HMM. The Baum-Welch algorithm is used to train the HMM. Only a few iterations are required to obtain stable values for  $\lambda_j(A_j, B_j, \pi_j)$ .

Once the HMM is trained, it describes the probability of observing certain hand posture sequences for the trained grasp type. It does not matter whether the user grasps the object quickly or slowly, as the state transitions are also modeled.

For classification, we select the most probable model, given the observations, i.e., the most probable grasp, given the sequence of hand postures. Since we do not have any *a priori* information about the models, this is the same as selecting the model with the highest likelihood of observing these symbols:

$$class = \operatorname{argmax}_i (P(\mathbf{o}_1 \dots \mathbf{o}_N | \lambda_i)) \quad (3)$$

3) *Mapping Human Grasps to Robot Grasps*: The human hand can perform a wide variety of grasps. Because of the difficulties in constructing a robotic gripper equivalent to the human hand, most robotic grippers have much less flexibility. However, the grippers may also have other features, that the human hand cannot compete with. Consequently, the human

and robot hand cannot, in general, perform the same types of grasps. A *mapping* of the grasp type from the human hand to the robot hand is necessary. If the task is intended to be executed by several different robot platforms, the mapping phase has to be done after the abstraction phase, as illustrated in Figure 1. However if the target robot platform is known, the grasp mapping may preferably be done at the same time as the grasp is recognized.

In this work, we have assumed the robot hand to be a Barrett hand with four degrees of freedom. We have restricted the system to three basic grasp types for the Barrett hand. These are *Wrap Grasp*, *Precision Grasp* and *Sphere Grasp*. The suitable grasp type depends both on the object and on the upcoming task. For example, tasks which require fine manipulation must make use of the *Precision Grasp*, while heavy objects, which require grasp stability, are best grasped with the *Sphere Grasp*. Cutkosky’s grasp hierarchy [12] organizes the human grasps into 16 different classes. Figure 5 illustrates how some of these grasp types map to the Barrett-hand equivalents.

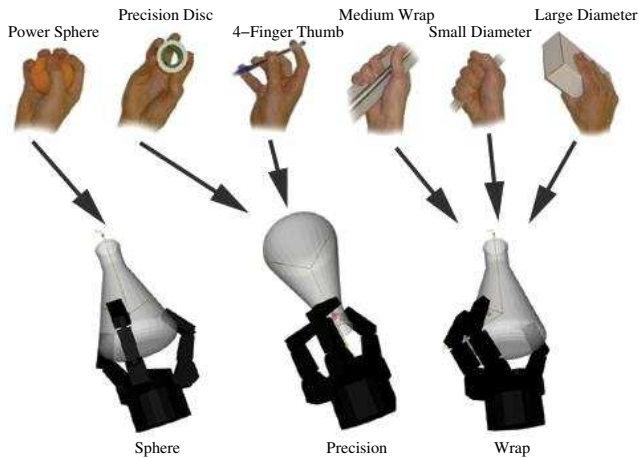


Fig. 5. Examples of the mapping from human grasp types to Barrett hand grasp types.

The mapping from human grasp to robot grasp is performed during training. The user demonstrates a grasp, and notifies the system to which of the three robot grasps his demonstration corresponds. It is not a problem that several human grasp types will correspond to the same robot grasp type, as the HMM will automatically account for this.

### B. Object Recognition

Objects are recognized by an appearance based method, using Color Cooccurrence Histograms (CCHs). The CCH is an extension of regular color histograms, and improves the recognition capabilities by capturing more of the texture features.

1) *Color Cooccurrence Histograms*: A CCH represents counts of how often *pairs* of pixels with certain colors occur in the image. One can either compute the histogram over the entire set of pixel pairs or constrain the number of pairs based on, for example, their relative distance. This way, only pixel

pairs separated by less than a maximum distance,  $d_{max}$  are considered. Using cross-validation, we have found that  $d_{max} = 10$  gives the best results. For a more detailed description of CCHs, see [13], [14].

2) *Object Recognition using Color Cooccurrence Histograms*: As the image typically contains many objects, it cannot be recognized directly by the CCH recognizer. Instead, the image is scanned with a small search window. At each window location, the CCH is calculated and compared with the searched object’s CCH. A vote, proportional to the histogram intersection, is stored, and when the entire image has been scanned, a vote matrix provides a hypothesis of the object’s location. Figure 6 shows a typical experimental scene and the corresponding vote matrix. In this case, the package of rice, in the upper right corner of the image, is being searched for. The vote matrix reveals a strong response in the vicinity of the object’s position (black colored square). Several smaller responses occur near the soda and cup, which contain similar colors. The process is repeated for each object that is being searched for.

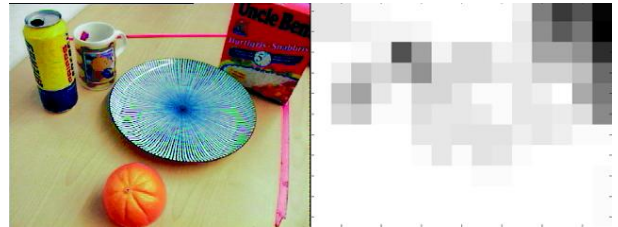


Fig. 6. The original image compared with the vote matrix for the orange rice packet.

3) *Object Localization*: Once the object has been recognized, its position is determined. The object center is calculated from the vote matrix in two steps. First, a rough segmentation of the object is performed, by starting from the strongest vote cell, and gradually expanding the borders. The process is described in detail in [13]. Then, the position is calculated as the average positions of all vote cells in this segmentation, weighted by their respective vote strength. The position is returned in screen coordinates by the object recognizer, and then transformed into world coordinates:

$$[w_x \ w_y \ 1] = [p_x \ p_y \ 1] \cdot M; \quad (4)$$

where  $\mathbf{w}$  denotes the world position,  $\mathbf{p}$  is the screen position in homogenous coordinates, and  $M$  is the  $3 \times 3$  camera transformation matrix, obtained by manually establishing four world-screen correspondences. Objects are assumed to be on a planar surface with known height, therefore  $w_z$  need not to be estimated.

### V. ACTION SEGMENTATION

As each action currently involves moving an object, the natural way to segment an action is to identify object movement. The action starts when an object is being moved, and is complete when object movement is no longer detected.

### A. Detecting Object Movement

Object movement can be detected by comparing the object’s location from frame to frame. However, the CCH based recognizer only provides a coarse location, and searching the whole image is quite time consuming. A way to speed up the process is to exploit the fixed mounted camera, and only look for temporal changes in the image. Before each subtask, a background image is stored. Then, during task demonstration, each pixel is compared with the background image, and only pixels which have changed color more than a certain threshold are kept. This reduces the background noise and also increases the recognition speed, as background pixels are not taken into account during recognition. Naturally, this means that some parts of the moving object will also be removed, as illustrated in Figure 7. However, this is only a minor problem when using CCHs, as the CCH does not rely on specific features that need to be visible. As we have shown in [13], the CCH recognition ratio is stable for noise values up to 25 %. Another drawback with the method is that the background behind the object will be visible when the object is moved from its initial position. Shadows is another issue, they may also reveal background, as seen in Figure 7. However, this method performs satisfactory for our purposes.

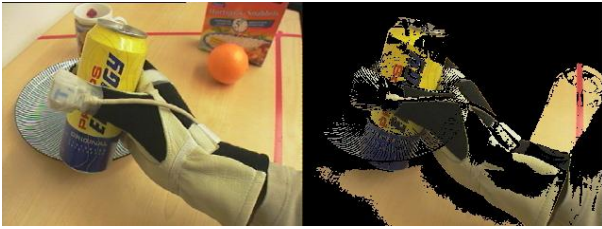


Fig. 7. Camera view of scene, before and after filtering with the background image.

### B. Grasp Segmentation

The start of an action is assumed to be when the user begins to grasp an object: before it is moved. We estimate the time point for grasp initiation by analyzing the hand posture data. As first suggested in [15], we estimate the optimal sequence length  $seqL$  by searching for the most likely grasp sequence in the past 20 posture changes. This search is initiated as soon as an object movement is detected, and is repeated for each grasp that we classify.

$$seqL = \arg \max_{i=minSeqL}^{maxSeqL} (a) \quad (5)$$

$$a = \max_{j=1}^{N_{grasps}} \underbrace{P(\mathbf{o}_{-i}, \dots, \mathbf{o}_0 | \lambda_j)}_{\text{Forward Procedure}} \cdot \underbrace{L^{-(maxSeqL-i)}}_{\text{Penalty term}}$$

where  $minSeqL$  and  $maxSeqL$  are the minimum and maximum sequence length for a grasp. As mentioned earlier,  $L$  are the number of possible observations and dimensions in the HMM-model. In our case, we had  $minSeqL = 5$ ,  $maxSeqL = 20$

and  $L = 50$ , since we have used 50 different posture clusters. Also,  $N_{grasps}$  are the number of possible grasp types to choose from, in our case 3.

In general, short sequences have higher likelihood than long ones. Equation 5 allows a weighted comparison between short and long sequences. Here, a penalty term that increases as the sequence length decreases is used. The penalty term represents the likelihood of observing a sequence of random  $\mathbf{o}$ , in addition to already available observations. It is assumed here that the hand posture likelihoods are uniformly distributed.

Once the sequence length has been established, the likelihood  $P_j$  for each model  $j$  is calculated as

$$P_j = P(\mathbf{o}_{-seqL}, \dots, \mathbf{o}_{-1}, \mathbf{o}_0 | \lambda_j) \quad (6)$$

## VI. EXPERIMENTAL EVALUATION

We have evaluated the system by demonstrating eight tasks, composed by a large variety of subtasks. For each subtask, the system reports what object is moved, where it is moved, and which robot grasp type is used.

### A. Task Composition

The current state of our system allows simple pick-and-place tasks to be programmed by demonstration. Five different objects are used: *cup*, *soda*, *plate*, *rice* and *orange*, shown in Figure 3. Currently, each task always starts with moving an object, the *reference object*, to the center. All other movements are considered relative to this location. An example of such a task is given in Table I. We settled with describing four relative locations: *behind*, *in front of*, *to the right of* and *to the left of*.

- |  |
|--|
| <ol style="list-style-type: none"> <li>1. Move plate with wrap grasp to center</li> <li>2. Move cup with precision grasp, behind plate</li> <li>3. Move rice with wrap grasp, to the right of plate</li> <li>4. Move orange with sphere grasp, to the left of plate</li> <li>5. Move soda with wrap grasp, behind plate</li> </ol> |
|--|

TABLE I

AN EXAMPLE OF A TASK PROGRAMMED BY DEMONSTRATION, USING OUR SYSTEM.

### B. Performance

The grasp recognition system was trained on the three grasp types. Each grasp type was demonstrated 30 times. Also, the object recognition system was trained with one segmented image per object. We tested the system by performing eight tasks, each involving five pick-and-place subtasks on five different objects. Table II shows the error rates on the three identifiers describing a subtask.

Of the total six wrong objects reported, five regarded the system mistaking the dataglove for the cup, because of the similar white color. The position errors occur because of the coarse position estimation used. It is the center, and not the base of the object, that is used for position estimation. Therefore, tall objects may appear further away from the

Wrong grasp type	15.0 %
Wrong object	15.0 %
Wrong position	12.5 %

TABLE II  
ERROR RATES FOR DIFFERENT TYPES OF ERRORS.

camera's point of view. The grasp type errors occur when the system confuses the sphere grasp and the wrap grasp. These two grasp types are quite similar for the human hand, but not for the Barrett hand. Figure 8 shows how the grasp type probabilities change during a grasp sequence. In this example, a wrap grasp is performed. As seen, all grasps are very improbable until the grasp is initiated and the hand begins to close. Once the grasp is complete, the wrap grasp has the highest likelihood. However, the grasp is not evaluated until object movement is detected, about one second later. In this time additional hand poses are detected, because of sensor noise, and the probabilities decrease. We plan to add touch sensors to the data glove, making it possible to evaluate the grasp right after it is complete.

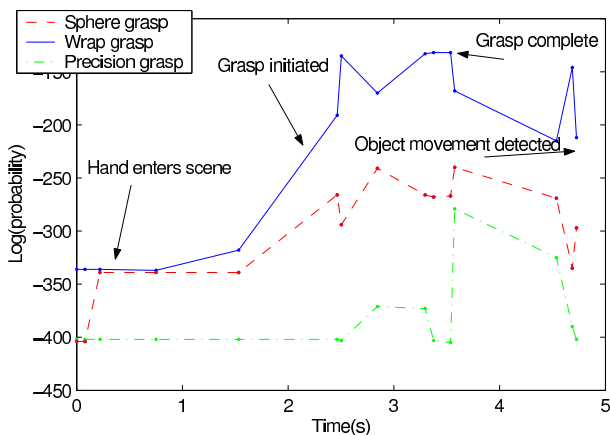


Fig. 8. The log probabilities for the different grasp types during a wrap grasp sequence. The probabilities are calculated in Equation 6

## VII. CONCLUSION

We have shown that pick-and-place tasks can be learned in a real environment. The images from the camera are analyzed using Color Cooccurrence Histograms, for object detection and localization. The grasp taken is recognized using magnetic trackers mounted on a glove, by the analyzing the hand pose transition sequence during the grasp, with Hidden Markov Models. Grasp mapping from human hand to robot hand may be done during HMM training.

Future work includes extending the system to learn more complex tasks, e.g. stacking of objects and dynamic grasps, such as *insert* or *screw*. Currently, only one demonstration is used for programming a task. As [5] point out, multiple demonstrations should be used for increased robustness and identification of irrelevant actions.

Additional data should be collected and stored during demonstration, which will aid the robot in executing the task. Instead of just learning what is being demonstrated, the robot may analyze the demonstration data in order to figure out how to replicate the demonstration. A promising approach to reduce the complexity of the sensor readings, is to map them onto primitives, [3]. The most suitable approach trajectory, given the objects' pose and grasp type, could then be estimated from the training examples obtained during demonstration. In the execution step, SIFT features [16] could be used to estimate the object pose, and the system could learn from multiple observations which grasp types are suitable for certain object rotations.

## REFERENCES

- [1] H. Friedrich, R. Dillmann, and O. Rogalla, "Interactive Robot Programming Based on Human Demonstration and Advice," in *Sensor Based Intelligent Robots*, pp. 96–119, 1998.
- [2] M. Ehrenmann, O. Rogalla, R. Zllner, and R. Dillmann, "Teaching service robots complex tasks: Programming by demonstration for workshop and household environments," in *Proceedings of the 2001 International Conference on Field and Service Robots(FSR)*, pp. 397–402, 2001.
- [3] D. C. Bentevegna and C. G. Atkeson, "Learning from observation using primitives," in *IEEE/RSJ International Conference on Robotics and Automation, ICRA'01*, 2001.
- [4] R. Dillmann, O. Rogalla, M. Ehrenmann, R. Zollner, and M. Bordegoni, "Learning robot behaviour and skills based on human demonstration and advice: the machine learning paradigm," in *9th International Symposium of Robotics Research, ISRR'99*, 1999.
- [5] K. Ogawara, J. Takamatsu, H. Kimura, and K. Ikeuchi, "Extraction of essential interactions through multiple observations of human demonstrations," *IEEE Trans. on Industrial Electronics*, vol. 50, 2003.
- [6] M. N. Nicolescu and M. J. Matarić, "Natural methods for robot task learning: Instructive demonstrations, generalization and practice," in *Proceedings of the Second International Joint Conference on Autonomous Agents and Multi Agent Systems*, 2003.
- [7] E. Lloyd, J. S. Beis, D. K. Pai, and D. G. Lowe, "Programming contact tasks using a reality-based virtual environment integrated with vision," *IEEE Trans. on Robotics and Automation*, vol. 15(3), 1999.
- [8] J. Aleotti, S. Caselli, and M. Reggiani, "Leveraging on a virtual environment for robot programming by demonstration," in *Robotics and Autonomous Systems, Special issue: Robot Learning from Demonstration*, vol. 47, pp. 153–161, 2004.
- [9] S. B. Kang and K. Ikeuchi, "Toward automatic robot instruction from perception - temporal segmentation of tasks from human hand motion," *IEEE Trans on Robotics and Automation*, vol. 11, pp. 670 – 681, October 1995.
- [10] K. Ogawara, K. Hashimoto, J. Takamatsu, and K. Ikeuchi, "Grasp Recognition using a 3D Articulated Model and Infrared Images," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2003.
- [11] L. Rabiner, "A tutorial on hidden Markov models and selected applications in speech recognition," in *Proc. of the IEEE*, vol. 77, no. 2, pp. 257–286, 1989.
- [12] M. Cutkosky, "On grasp choice, grasp models and the design of hands for manufacturing tasks," *IEEE Transactions on Robotics and Automation*, 1989.
- [13] S. Ekvall, F. Hoffmann, and D. Kragic, "Object recognition and pose estimation for robotic manipulation using color cooccurrence histograms," in *IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS'03*, 2003.
- [14] P. Chang and J. Krumm, "Object recognition with color cooccurrence histograms," in *CVPR'99*, pp. 498–504, 1999.
- [15] S. Ekvall and D. Kragic, "Interactive grasp learning based on human demonstration," in *IEEE/RSJ International Conference on Robotics and Automation, ICRA'04*, 2004.
- [16] D. Lowe, "Object recognition from local scale-invariant features," in *International Conference on Computer Vision*, pp. 1150–1157, 1999.