The 2010 IEEE/RSJ International Conference on
Intelligent Robots and Systems
October 18-22, 2010, Taipei, Taiwan

Robot Learning of Everyday Object Manipulations via Human Demonstration

Hao Dang and Peter K. Allen

***Abstract*— We deal with the problem of teaching a robot to manipulate everyday objects through human demonstration. We first design a task descriptor which encapsulates important elements of a task. The design originates from observations that manipulations involved in many everyday object tasks can be considered as a series of sequential rotations and translations, which we call manipulation primitives. We then propose a method that enables a robot to decompose a demonstrated task into sequential manipulation primitives and construct a task descriptor. We also show how to transfer a task descriptor learned from one object to similar objects. In the end, we argue that this framework is highly generic. Particularly, it can be used to construct a robot task database that serves as a manipulation knowledge base for a robot to succeed in manipulating everyday objects.**

I. Introduction

Having robots that are capable of doing meaningful jobs in human environments, such as homes, offices, or restaurants, has been attracting an increasing amount of attention from researchers. Not only does it improve the quality of life but it also frees people from many chores. Several researchers have proposed methods from different perspectives for robots to open doors [1], [2], [3], [4], [5]. A group consisting of researchers from Intel and CMU built *Herb* [6], which is a butler robot in a home environment. Other similar robots are the ARMAR humanoid robot [7], the HRP-2 robot [8], and the Care-o-bot II [9]. Although these works have provided robots with more intelligent capabilities, they require explicit programming by robotics experts and they are pre-designed for specific tasks. This inherently leaves adaptability of these capabilities a problem when a robot works in different environments. The reasons for this are threefold. First, it is often difficult or even impossible to take into account all the possible environments robots can work in during the initial design. Second, specifications of tasks, even similar ones, can vary dramatically and it is difficult to satisfy different task specifications in a pre-programmed manner. Third, with the increasing use of robots, the overwhelming number of different tasks required for robots makes it impossible for experts to program every one of them.

To solve this problem, we need a more generic way to generate robot skills which are adaptive to different working environments. This solution should have an internal generic representation of tasks that is comprehensible to robots and transferable to basic physical robot manipulations. It needs to provide a way for automatic robot learning to transfer human knowledge into an internal generic representation. It should be easy for non-experts to construct robot skills and it has to be generic enough for long-term robot learning and evolution. Programming by demonstration (PbD) is a promising approach to this end. Instead of programming a robot through machine commands, the PbD technique allows end users to teach robots new behaviors by demonstrating them in the end-user environment. In this framework, robot learning benefits from the following aspects. First, end users are most knowledgeable about the exact tasks robots are intended to accomplish. Second, robot skills are generated in the exact final working environment. These factors make the robot learning under the PbD framework more effective and efficient. In this realm, Calinon *et al.* presented a PbD framework for representing and generalizing a task [10]. Dillmann developed an approach for teaching a robot typical tasks required in everyday household environments [11]. Rusu *et. al.* built a robot that operates autonomously within a sensor-equipped kitchen [12]. Ekvall and Kragic proposed techniques for task level learning and planning [13]. Kober *et. al.* [14] and Theodorou *et. al.* [15] used machine learning techniques to teach robots motor skills. Other related works include [16], [17], [18], and [19].

Most of the work discussed above is either at a low level (i.e. at motor control layer), or a high level, (i.e. at logic reasoning layer). A notable exception is the work of Prats *et al.* [20]. However, there is still a gap between these two different levels. On the one hand, in order to improve robot flexibility and adaptability, a generic task representation is needed. This creates a connection upwards to a high-level task description. On the other hand, a technique to connect with a low-level physical motor movement is needed for the execution of the high-level task description.

Our work is at the middle layer where we design a framework that is feasible for the high-level task abstraction and the low-level motion execution. In Section II, we propose a design of a task descriptor. In Section III and IV, we discuss about how to obtain each element of a task descriptor. In Section V, we describe how a task is executed and how a task descriptor learned on one object can be transferred to similar objects. Experiments on everyday object manipulations are shown in Section VI followed by conclusions in Section VII.

II. Design of A Task Descriptor

In this section, we first show a collection of everyday object manipulations and then conclude with a design of a task descriptor.

Hao Dang and Peter K. Allen are with Department of Computer Science, Columbia University, 450 Computer Science Building, 1214 Amsterdam Avenue, New York, NY, 10027, USA, {dang,allen}@cs.columbia.edu

A. Observations

The design of a task descriptor is based on observations of tasks on everyday objects, such as opening doors, opening drawers, opening bottle caps, etc. Fig. 1 shows a series of these everyday objects. A human hand is the end effector in a task. The trajectory of a human hand during a manipulation provides most of the information of this task. Considering the fact that during a task execution the human hand is relatively still with respect to the object, we can say the motion of rigid parts of the object also carries equivalent information. This information tells us what motions are involved to accomplish the task. By closely examining motions in these tasks, we can see that most of them can be modeled as a chain of rotations and translations. In Fig. 1, the rotation axis and the translation direction are illustrated with solid black lines. For example, opening a door includes two rotations, along axis I and along axis II as shown in Fig. 1(f).

B. Manipulation Primitive

Based on the observations above, a task on an everyday object can be decomposed into a chain of sequential manipulations represented by rotations and translations. This is related to the kinematic models of the objects [21], [22] or related motion between rigid bodies [23]. This manipulation chain is the most important part in describing a task. It specifies the motions of each step. We call each of these rotations or translations a manipulation primitive. We define manipulation primitive as follows.

***Definition** 1 (Manipulation Primitive):* A manipulation primitive is a pair of vectors,

$$P = \langle pos, dir \rangle \tag{1}$$

where

- $pos = [p_x, p_y, p_z]$ is a point in 3D space where the rotation axis is located. It is ignored in the translation case.
- $dir = [d_x, d_y, d_z]$ is a direction for both rotation and translation. In rotation, it denotes the direction of the rotation axis. In translation, it denotes the translation direction.

As an example, a door-opening task consists of two sequential rotation manipulation primitives as is shown in Fig. 1(f). Both of these rotation axes need to be defined in the 3D door coordinate system so that a robot can transfer this chain of notations to a chain of physical motions.

C. Task Descriptor

Based on manipulation primitives above, we can now synthesize a task. We give the following definition for a task descriptor.

***Definition** 2 (Task Descriptor):* A task descriptor $\mathcal{T}$, is a pentuple

$$\mathcal{T} = \langle \mathcal{O}, \mathcal{A}, \mathcal{G}, \mathcal{M}, \mathcal{C} \rangle \tag{2}$$

where

- $\mathcal{O}$ refers to the 3D object geometry, e.g. a scanned door handle model.
- $\mathcal{A}$ is an action identifier, e.g. "open".
- $\mathcal{G}$ denotes a representation of the grasp in the demonstration, e.g. an eigengrasp [24].
- $\mathcal{M} = \{P_1, P_2, ..., P_k\}$ is a manipulation chain consisting of k manipulation primitives, where $P_i, i \in [1, k]$ denotes the i-th manipulation primitive (e.g. $\mathcal{M}$ for opening a door should contain two manipulation primitives. One is a rotation on the door handle. The other is a rotation around the door hinge.)
- $\mathcal{C}$ is used for extra information such as task constraints and transition conditions. The transition conditions define when the robot should shift from an execution of the current manipulation primitive to the next. In our implementation, $\mathcal{C}$ stores the moving distance along the translation direction or the rotation angle around the rotation axis demonstrated by the end user.

III. Learning A Manipulation Chain

A task can contain more than one manipulation primitive, combined together as a manipulation chain $\mathcal{M}$ inside a task descriptor $\mathcal{T}$. This section deals with the problem that given a demonstration by a human, how can a robot automatically learn from this demonstration the sequential manipulation primitives and construct a manipulation chain?

A. Human Demonstration

A human demonstration should be as natural as possible. During the demonstration, the robot should be able to see how the human is manipulating the object and reason based on what it sees or records. In our problem, we assume what the robot sees is the motion of discrete points on the object and the human hand. If these points sample the object in a way such that they cover every rigid part of the articulated object, the information of how the object is manipulated is embedded in the trajectories of those visible points. Considering that the human hand is relatively still to the object as mentioned in Sec. II-A, the trajectory of the human hand also contains the information of the manipulation.

B. Learning A Translation

Given a trajectory of one marker with n records, say $Tr = \{p_1, p_2, ..., p_n\}$, where $p_i = [x_i, y_i, z_i]^T$, assume they are all on one individual translation axis. We want to extract the translation axis ℓ_t out of this trajectory Tr. We apply PCA analysis on these records and find out the direction of the largest covariance. This direction corresponds to the translation axis. We have $dir = \ell_t = \boldsymbol{e_{max}}$ where $\boldsymbol{e_{max}}$ is the eigenvector of the eigenvalue with the largest magnitude.

Because translation is just a direction, we do not need a specific point to locate the origin of the motion. A manipulation primitive learned for a translation only uses the dir part in P and ignores the pos part, which is $P_{Transl} = \langle undefined, \boldsymbol{e_{max}} \rangle$

C. Learning A Rotation

Given a trajectory of one marker with n records, say $Tr = \{p_1, p_2, ..., p_n\}$, assume they are produced by a rotation of

(a) speaker button (b) door handle (c) bottle cap (d) key (e) switch (f) whole door

Fig. 1. Observations on everyday object manipulations. Axis of rotations or translations are annotated in the images.

this marker around an axis ℓ_r. We want to locate this axis to specify our manipulation primitive P_{Rot}. To specify this axis, we need six parameters. We first calculate the direction of the axis, dir, and then locate the axis by calculating a point, pos, through which the axis passes.

Considering a marker rotating around one fixed axis, we know ideally all the points on its trajectory should be on one plane, which is orthogonal with respect to the rotation axis. Thus, we know for point locations in this trajectory, there must exist one dimension where the variance of points is significantly small, theoretically zero. This dimension corresponds to the eigenvector of the eigenvalue of the smallest magnitude. Then when we apply PCA analysis on a trajectory of a rotation, the eigenvector corresponding to the eigenvalue of the smallest magnitude is the rotating axis. We have $dir = \boldsymbol{e_{min}}$ where $\boldsymbol{e_{min}}$ is the eigenvector of the eigenvalue with smallest magnitude.

We need to determine a point in space, pos, where this axis passes through. This is done by circle fitting in the plane, which is spanned by the eigenvectors of the two largest eigenvalues in magnitude, to locate the center in this plane and reverse projecting it to the original space.

D. A Translation or A Rotation

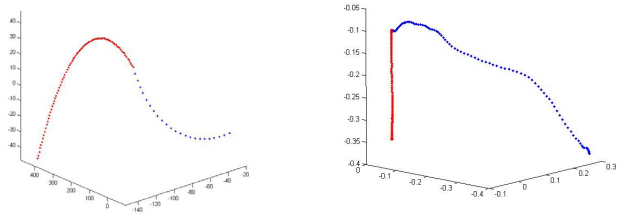
Given a trajectory, we need to decide whether it is a translation or a rotation before we apply any analysis described above. If it is a rotation, the points should be on a plane spanned by two orthogonal directions. If it is a translation, the points should be on a single line spanned by one direction. So, we apply PCA analysis on the trajectory and compare the magnitude of the three eigenvalues. We define a threshold $T_0 \in [0, 1]$ to distinguish rotation from translation via the following function. We assume the magnitude of the eigenvalues are already ordered such that $\lambda_1 \geq \lambda_2 \geq \lambda_3$, where λ_i is the magnitude of the i-th eigenvalue.

$$Motion = \begin{cases} rot & \text{if} \quad \lambda_3 < T_0 S \ \& \ \lambda_2 > T_0 S \\ transl & \text{if} \quad \lambda_1 > (1 - T_0) S \\ undefined & \text{if} \quad otherwise \end{cases} \tag{3}$$

where $S = \sum_{i=1}^{3} \lambda_i$.

E. Learning A Manipulation Chain

The raw data observed by a robot is a sequence of 3D locations of markers. These trajectories can contain multiple manipulation primitives and we need to segment records from different manipulation primitives.



(a) Opening a rotating door. (b) Opening a sliding door.

Fig. 2. Trajectory Segmentation

TABLE I
TRAJECTORY SEGMENTATION

Input: $\{(p_1, t_1), ...(p_i, t_i), ...(p_n, t_n)\}$, where $p_i = [x_i, y_i, z_i]$ and k
Output: $\{k_1, ...k_i, ...k_n\}$, where $k_i \in [1, k]$

1. compute affinity matrix A as $A_{ij} = \exp\left(-||p_i - p_j||^2/2\delta_p^2 - ||t_i - t_j||^2/2\delta_t^2\right)$
2. construct matrix $L = D^{\frac{1}{2}} A D^{-\frac{1}{2}}$ where $D_{ii} = \sum_{j=1}^{N} A_{ij}$
3. compute matrix $E = [e_1, ..., e_k]$, where e_i is normalized first k largest eigenvectors of L
4. treat each row of E as a point in R^k, cluster them into k clusters by K-Means clustering algorithm
5. assign the original point to cluster i if row j of E was assigned to cluster i

1) Trajectory Segmentation: Given a trajectory of n records, $Tr = \{(x_1, y_1, z_1, t_1), ..., (x_n, y_n, z_n, t_n)\}$, the trajectory segmentation algorithm divides it into k segments such that each segment corresponds to one individual manipulation primitive. We use spectral clustering [25] to do the segmentation. In Table I, we illustrate the algorithm in detail. Fig. 2 shows several segmentation results based on this algorithm. The parameter k is set by the end user depending on how many different segments that may exist in the demonstrated trajectory.

2) Constructing A Manipulation Chain: Based on the trajectory segmentation result, we have divided the whole trajectory into k segmented trajectories, i.e. $Tr = \{Tr_1, Tr_2, ...Tr_k\}$. Each of these k trajectories is from one manipulation primitive. We apply our analysis as described in Sections III-B and III-C to each trajectory Tr_i and get the corresponding manipulation primitive P_i. Putting them together in their original temporal order, we can construct a manipulation chain $\mathcal{M} = \{P_1, P_2, ..., P_k\}$. Fig. 4 shows the results of the learned manipulation primitives. Sometimes, a trajectory is in fact neither a rotation nor a translation, e.g. the hand approach trajectory in Fig. 2(b). In these cases the

motion is classified as undefined and the algorithm will delete this trajectory and eliminate it from the final manipulation chain.

IV. Grasp Planning

Once the task primitives and manipulation chain have been learned, we need to compute a grasp on the object for the task. This is an example of task-oriented grasp planning [26] [27]. This approach takes into account the specific task and hence may result in a more effective grasp for the task. Adaptability is an important aspect in object manipulation and task-oriented grasp planning is important for robots working in environments which are constantly changing.

To compute our grasp, we use knowledge learned from the human demonstration of the task. During the learning phase, we can track the human wrist position using a 6 DOF pose sensor (e.g. Flock-of-Birds magnetic sensor). This provides us with an approach and orientation vector for the grasp. To select the actual grasp with the robotic hand, we cannot use the human grasp finger positions as our robotic end-effector will have different kinematics than a human hand.

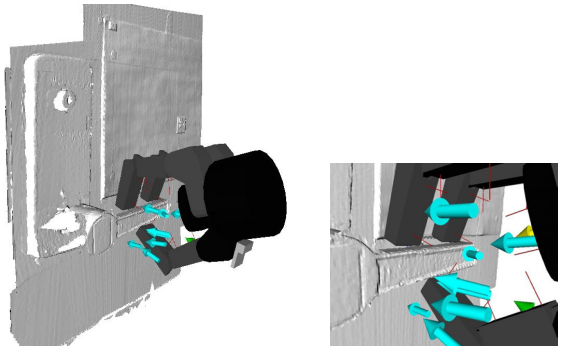
A solution is to use our online $eigengrasp$ grasp planner which is a part of our *GraspIt!* simulator [24], [28]. This planner takes the wrist position of the hand $\mathcal{V}$ and a model of an object to be grasped $\mathcal{O}$ and computes a stable grasp. It works in real-time by using $eigengrasps$ which dramatically reduce the dimensionality of the search for a stable grasp.

We now describe our method for the example of opening a door (see figure 3). The user's wrist position is tracked in 3D space during the learning phase using the Flock-of-Birds sensor. To generate an object model, we use a Microscribe 3D digitizer to get a model of the handle. The on-line eigengrasp planner then plans a stable grasp using the wrist position of the robotic hand specified and the modeled object $\mathcal{O}$. In our case, the physical robot hand used is a Barrett hand. This grasp is stored in the task descriptor and used by the task execution module which executes the learned task.

This user-interactive grasp planning approach allows us to combine the user's prior knowledge on the selection of good grasp locations and the power of the eigengrasp planner. Since the process is interactive and real-time, users can modify the grasp candidate as they change approach directions. For the grasp representation $\mathcal{G}$ we use the first two eigengrasps which we store along with the 6 DOF of the wrist position as $\mathcal{G} = [a_1, a_2, wrist\_pos]$ where a_1 and a_2 are the amplitudes along the first two eigengrasp directions and $wrist\_pos \in \mathbb{R}^6$ specifies the extrinsic DOFs to determine the position and orientation of the wrist. This very compact representation makes the task extensible to other robotic hands, and allows us to transfer this learned grasp from one robotic hand to another.

V. Task Execution and Transfer

We discussed how to learn a task and construct the corresponding task descriptor from a human demonstration. With such a task descriptor, a robot can execute this learned task. Also, this task descriptor allows us to transfer tasks learned from one object to similar objects.



(a) Planned grasp (b) A closeup

Fig. 3. Grasp planning using the on-line grasp planner

A. Task Execution

Given a required task $\mathcal{T}$, which is associated with an action $\mathcal{A}$ and an object $\mathcal{O}$, the robot first registers the object in the real world with its internal geometry representation of the object $\mathcal{O}$. It then goes to the position specified by $\mathcal{G}$ and applies this grasp. After this, the robot generates the required motion of the hand according to the manipulation chain $\mathcal{M}$ and executes the task.

B. Task Transfer

Consider opening a bottle cap as illustrated in Fig. 1(c) as an example. We want to apply a task descriptor learned on it to similar bottles. We formalize this problem as follows: given a task descriptor $\mathcal{T}$ and an object $\mathcal{O}'$ which is geometrically similar to $\mathcal{O} \in \mathcal{T}$, how can a robot build a task descriptor $\mathcal{T}'$ on $\mathcal{O}'$ which has the same semantic meaning as $\mathcal{T}$? To transfer a task descriptor $\mathcal{T}$ to $\mathcal{T}'$, we need to transfer the motion primitives $\mathcal{M} \in \mathcal{T}$ as well as the grasp $\mathcal{G} \in \mathcal{T}$ and construct $\mathcal{M}'$ and $\mathcal{G}'$.

The transformation can be obtained by aligning the two objects. We use the Iterative Closest Point algorithm (ICP) [29] to compute the transformation between the two objects $\mathcal{O}$ and $\mathcal{O}'$. Let Tr be the transformation from $\mathcal{O}$ to $\mathcal{O}'$. Then, generating the new motion primitives can be done by applying Tr to the old ones. The corresponding location and orientation of the new grasp can be computed in the same way. Based on this, a new task descriptor is constructed. We note that this is dependent on how good the alignment is. In the following section, we show an example of transferring a bottle-cap-opening task.

VI. Experiments and Initial Results

A. Experiment setups

We have done three experiments for task execution and one experiment for task transfer. During human demonstration, a motion capture system (optiTrack) was used to record 3D marker trajectories. These markers were placed on the experimental objects by the end user. Trajectories of these markers were then fed as input to the manipulation chain learning algorithm discussed in Section III. For grasp planning, a Flock-of-Birds sensor was used for locating hand wrist. An accurate geometrical model of the object was obtained with a Microscribe 3D digitizer and was provided to the grasp

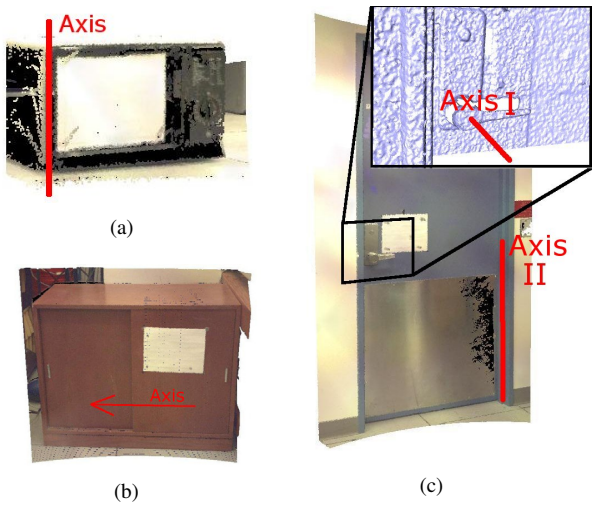


Fig. 4. 3-D objects with the manipulation primitive annotations.

planner discussed in section IV. Once we had learned the manipulation chain and computed the grasp, we were able to effect task execution. In the task execution, we again used the 3D digitizer to manually register the object with the robot's coordinate system and the motion capture system.

B. Opening a microwave - Learning a rotation

In this experiment, three markers were placed on the door of the microwave. Fig. 4(a) shows a texture-mapped 3D model of the microwave with the rotation axis learned by our algorithm overlaid on it. Fig. 5(a) to 5(e) show the task execution when a robot rotated the microwave door open.

C. Opening a sliding door - Learning a translation

In the PbD phase, four markers were placed on the sliding door and one marker was placed on the thumb of the user. Markers on the thumb have two major trajectories. The first part corresponds to the reaching action. The second part corresponds to the sliding action. Fig. 2(b) shows the segmentation of the trajectory of a marker on the thumb. Fig. 4(b) shows the 3D model with the translation axis learned by our algorithm. Fig. 5(f) to 5(j) show the task execution when a robot slid the cabinet door open according to the task descriptor learned.

D. Opening a door - Learning two rotations

In the PbD phase, three markers were placed on the door and the handle. Markers on the handle have two main trajectories, i.e. one rotation along the handle hinge and another along the door hinge. Fig. 2(a) shows the segmentation of the trajectory of a marker on the door handle. Fig. 4(c) shows the 3D model with the rotation axes learned by our algorithm. We did not do task execution on this since the door is not within the working space of the robot.

E. Task Transfer - Transfer a bottle-cap-opening task

Opening the left bottle in Fig. 5(k) was performed by the user and the corresponding task descriptor was constructed. This experiment was meant to transfer this task descriptor to another similar but novel bottle, which is on the right in Fig. 5(k). The digitizer was used to model the two bottle caps. The method discussed in Section V-B was applied to align the two bottle caps and transfer the learned task descriptor to the new bottle. We did not use the whole bottle data for the alignment, because only the bottle caps are the functional part of the bottle during the task. There is a 6-degree offset between the transferred rotation axis and the ideal axis. The axis passes through the bottle cap at a distance of 4mm from its surface center. Fig. 5(l) to 5(o) show an experiment when a robot executes this new task descriptor to open the novel bottle. We see the error in the alignment was accurate enough and the task was successfully executed.

VII. Conclusion and Future Work

In this paper, a framework for *Robot Learning by Demonstration* is proposed. This framework goes from a high level task abstraction to a low level motion generation. First, a generic task descriptor is designed based on observations of everyday object manipulations. Then, a method for robots to automatically learn from human demonstration and construct such task descriptors is developed. In our experiments, we showed results for this design and implementation.

Although the task descriptor proposed in this paper cannot model all the tasks in our everyday life, it works for many of those we encounter. To make it more generic, we will be working on more motion primitives, e.g. B-splines and spiral curves and more comprehensive representation of the task constraints as in $\mathcal{C}$. We argue that this framework is extendable by taking advantage of recent shape matching advances in robot manipulation. Considering different objects with similar functions, such as different door handles in Fig. 1(d) and 1(f), we should expect them to share very similar geometries - they all have a horizontal bar and it can be rotated. With shape matching techniques, task descriptors can be transfered between objects with similar geometries. This makes our framework extendable to a more general case where task descriptors are generated based on one specific object but are capable of being applied to many other similar objects. In Section VI-E, we showed an experiment of task transfer on opening a bottle cap. As we mentioned in Section V-B, task transfer relies on good alignment between objects. Our current ICP method does not handle scaling or shearing between objects very well. To alleviate this problem, we are currently implementing advanced shape matching and alignment techniques from our previous work [30]. In the trajectory segmentation algorithm, the parameter k is set by the user. In the future, we will be working on algorithms to make this process automatic. One solution is using techniques developed by Lihi Zelnik-manor and Pietro Perona [31]. In the task execution phase, we registered the object to the robot manually and we will also be working on automating this process using vision techniques.

VIII. Acknowledgements

The authors would like to thank Professor Steven Feiner and Nicolas Dedual for sharing their motion capture system.

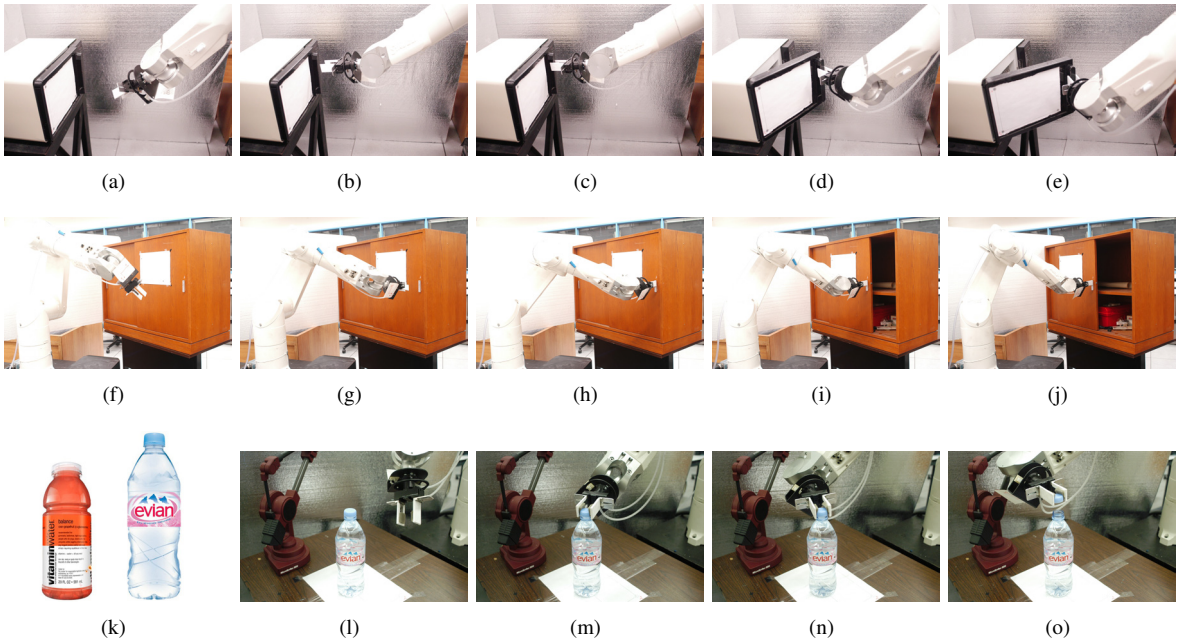


Fig. 5. Experiments. 5(a) to 5(e) show the robot opening a microwave. 5(f) to 5(j) show the robot sliding open a cabinet. 5(l) to 5(o) show the robot rotating open the bottle cap based on the task descriptor transfered from the bottle-cap-opening task learned on the vitamin bottle (left in 5(k)).

REFERENCES

[1] E. Klingbeil, A. Saxena, and A. Y. Ng, "Learning to open new doors," in *RSS Workshop on Robot Manipulation*, 2008.

[2] A. Jain and C. C. Kemp, "Pulling open novel doors and drawers with equilibrium point control," in *IEEE-RAS Intl. Conf. on Humanoid Robotics (Humanoids)*, 2009.

[3] D. Anguelov, D. Koller, E. Parker, and S. Thrun, "Detecting and modeling doors with mobile robots," in *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2004, pp. 3777–3784.

[4] D. Kragic, L. Petersson, and H. I. Christensen, "Visually guided manipulation tasks," vol. 40, no. 2-3, pp. 193 – 203, 2002.

[5] G. Niemeyer and J.-J. Slotine, "A simple strategy for opening an unknown door," in *Robotics and Automation, 1997. Proceedings., 1997 IEEE intl. conf. on*, vol. 2, 1997, pp. 1448 –1453 vol.2.

[6] S. Srinivasa, D. Ferguson, C. Helfrich, D. Berenson, A. Collet, R. Diankov, G. Gallagher, G. Hollinger, J. Kuffner, and M. VandeWeghe, "Herb: A home exploring robotic butler," 2009.

[7] T. Asfour, K. Berns, and R. Dillmann, "The humanoid robot armar: Design and control," in *Intl conf. on Humanoid Robots*, 2000.

[8] K. Kaneko, F. Kanehiro, S. Kajita, H. Hirukawa, T. Kawasaki, M. Hirata, K. Akachi, and T. Isozumi, "Humanoid robot hrp-2," in *IEEE Int. Conf. Rob. Aut*, 2004, pp. 1083–1090.

[9] B. Graf, M. Hans, and R. Schraft, "Care-o-bot ii-development of a next generation robotic home assistant," vol. 16, pp. 193–205, 2004.

[10] S. Calinon, F. Guenter, and A. Billard, "On Learning, Representing and Generalizing a Task in a Humanoid Robot," vol. 37, no. 2, pp. 286–298, 2007.

[11] R. Dillmann, "Teaching and learning of robot tasks via observation of human performance," vol. 47, no. 2-3, pp. 109 – 116, 2004, robot Learning from Demonstration.

[12] R. B. Rusu, B. Gerkey, and M. Beetz, "Robots in the kitchen: Exploiting ubiquitous sensing and actuation," 2008.

[13] S. Ekvall and D. Kragic, "Robot learning from demonstration: A task-level planning approach," 2008.

[14] J. Kober and J. Peters, "Learning motor primitives for robotics," in *IEEE Intl. Conf. on Robotics and Automation*, 2009.

[15] S. S. Evangelos A. Theodorou, Jonas Buchli, "reinforcement learning of motor skills in high dimensions: a path integral approach." in *intl. conf. of robotics and automation (icra 2010)*, 2010.

[16] C. L. Campbell, R. A. Peters, R. E. Bodenheimer, W. J. Bluethmann, E. Huber, and R. O. Ambrose, "Superpositioning of behaviors learned through teleoperation," vol. 22, pp. 1–13, 2004.

[17] O. C. Jenkins and M. J. Matarić, "Performance-derived behavior vocabularies: Data-driven acqusition of skills from motion," vol. 1, pp. 237–288, Jun. 2004.

[18] D. Bentivegna and C. Atkeson, "Learning from observation using primitives," in *Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE Intl. Conf. on*, vol. 2, 2001, pp. 1988–1993.

[19] K. Ogawara, J. Takamatsu, S. Iba, T. Tanuki, H. Kimura, and K. Ikeuchi, "Acquiring hand-action models in task and behavior levels by a learning robot through observing human demonstrations," in *The IEEE-RAS Intl. Conf. on Humanoid Robots*, Sep. 2000.

[20] M. Prats, P. Sanz, and A. del Pobil, "A framework for compliant physical interaction," *Autonomous Robots*, 2009.

[21] J. Sturm, K. Konolige, C. Stachniss, and W. Burgard, "Vision-based detection for learning articulation models of cabinet doors and drawers in household environments," in *Proc. of the IEEE Int. Conf. on Robotics and Automation*, 2010.

[22] D. Katz, Y. Pyuro, and O. Brock, "Learning to manipulate articulated objects in unstructured environments using a grounded relational representation," in *Proceedings of Robotics: Science and Systems IV*, Zurich, Switzerland, June 2008.

[23] J. Morrow and P. Khosla, "Manipulation task primitives for composing robot skills," in *Robotics and Automation, Proceedings., IEEE International Conference on*, 1997, pp. 3354 –3359.

[24] M. T. Ciocarlie and P. K. Allen, "Hand Posture Subspaces for Dexterous Robotic Grasping," vol. 28, no. 7, pp. 851–867, 2009.

[25] A. Y. Ng, M. I. Jordan, and Y. Weiss, "On spectral clustering: Analysis and an algorithm," in *Advances in Neural Information Processing Systems 14*, 2001, pp. 849–856.

[26] Z. Li and S. Sastry, "Task oriented optimal grasping by multifingered robot hands," in *IEEE Intl. Conf. on Robotics and Automation*, 1987, pp. 389–394.

[27] R. Haschke, J. Steil, I. Steuwer, and H. Ritter, "Task-oriented quality measures for dextrous grasping," in *Computational Intelligence in Robotics and Automation, 2005. CIRA 2005. Proceedings. 2005 IEEE intl. Symposium on*, June 2005, pp. 689–694.

[28] A. T. Miller and P. K. Allen, "Graspit! a versatile simulator for robotic grasping," vol. 11, no. 4, pp. 110–122, 2004.

[29] Z. Zhang, "Iterative point matching for registration of free-form curves and surfaces," vol. 13, no. 2, pp. 119–152, 1994.

[30] C. Goldfeder, M. T. Ciocarlie, J. Peretzman, H. Dang, and P. K. Allen, "Data-driven grasping with partial sensor data." in *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ intl. conf. on*. IEEE, 2009, pp. 1278–1283.

[31] L. Zelnik-manor and P. Perona, "Self-tuning spectral clustering," in *Advances in Neural Information Processing Systems 17*. MIT Press, 2004, pp. 1601–1608.