Learning by Observation with Mobile Robots: A Computational Approach

Kevin R. Dixon and Pradeep K. Khosla
Electrical & Computer Engineering
Carnegie Mellon University
Pittsburgh, PA 15213
Email: {krd, pkk}@cs.cmu.edu

***Abstract*— We present a computational approach to Learning By Observation (LBO) that allows users to program mobile robots by demonstrating a task. Unlike previous approaches, our system incorporates statistical-learning techniques and concepts from control theory to reduce the amount of domain knowledge needed to infer the intent of the user. To improve the generalization ability of the system, the user can demonstrate the task multiple times. We extract task subgoals from these demonstrations and automatically associate them with objects in the environment. As these objects move, the subgoals are updated accordingly. This gives our system the ability to learn from demonstrations performed in different environments. In this paper, we present the concepts used in our LBO system as well as experimental laboratory results in learning motor-skill tasks.**

I. Introduction and Related Work

Despite the numerous advances in human-robot interaction, most development systems still require that users have substantial knowledge of procedural-programming techniques as well as the specific robot system at hand. For the vast majority of the population, this effectively precludes the use of robots in most cases. If robots are to make headway into everyday situations, then users must be able to program robots in a more natural and intuitive manner.

Simplifying the interaction between humans and robots has received an increasing amount of attention recently and there have been a wide variety of approaches pursued. Iba et al. [1] have developed a system that incorporates speech and gesture, instead of a keyboard and joystick, to interact with a vacuum-cleaning mobile robot. This type of system targets users who are experts at a particular task but may have limited programming ability. Several researchers have also developed systems that observe users demonstrating tasks and synthesize the information so that a robot can perform the tasks and achieve the desired goal. These systems go by several names including Learning By Observation (LBO). The best-known example of LBO is the neural-network system, ALVINN, that learns to drive a car by observing human drivers [2]. One of the hallmarks of LBO is a scarcity of data on which to train the system since the demonstrations are the result of human activity, which may be of significant duration. As a result, many researchers decompose demonstrations into a symbolic representation and incorporate extensive domain knowledge to bias learning. Chen and Zelinsky [3] use a symbolic configuration-space description to represent a simple assembly task. Suboptimal human demonstrations were ameliorated by "filtering" the observations and by incorporating extensive domain knowledge with heuristic methods. By relying heavily on task-specific information this approach would require reprogramming for each new target task. Nicolescu [4] created a system that decomposes a demonstration into a symbolic set of robotic behaviors. Using dynamic programming and human feedback, a deterministic "behavior network" is constructed from multiple examples of a task. This allows the system to extract user intentions from a relatively small training set. Ijspeert et al. [5] have developed a computational method for movement imitation that can incorporate multiple demonstrations of a task. This system can also modulate the learned trajectory based on objects in the environment. Due to its internal representation, this approach only allows a single scaling or shifting parameter applied to the entire trajectory so that if only some objects in the environment move, then complete retraining would be required.

Fig. 1. The mobile robot used in these experiments, Agent Orange.

User demonstrations can be viewed as a sequence of goal-directed actions. Consider a user teaching a robot to vacuum a room by walking through the desired route. It is undesirable to require retraining each time furniture is moved. By capturing the *intent* of the user, an LBO system can determine the sequence of subgoals and overall goal of the task despite changes in the environment, human imprecision, or sensor noise. The

To appear in Proceedings of the IEEE International Conference on Robotics and Automation
April 26 - May 1, 2004, New Orleans, Louisiana USA

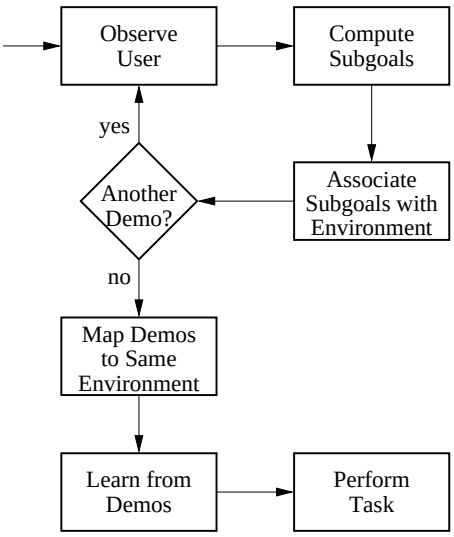


Fig. 2. Conceptual flow diagram of Dollop.

objective of our work is to create an LBO system that reduces the dependence on *a priori* task-specific information while producing robust behavior. To this end, we have created an LBO system called "Dollop" that employs statistical-learning techniques, as well as concepts from control theory, to allow users to program a mobile robot through one of the most natural methods possible: walking around. The target tasks for Dollop are motor-learning skills where the demonstrations may occur in different environments. We show that a computational approach to LBO can extract user intentions accurately in laboratory experiments. The conceptual flow diagram of Dollop is shown in Figure 2. While observing the user, Dollop extracts important points, or subgoals, from the trajectory. These subgoals are automatically associated with objects in the environment so that as the environment changes, the subgoals are updated accordingly. When learning from multiple demonstrations performed in different environments, Dollop hypothesizes how the user would have performed the task under the same conditions. A learning algorithm then estimates a statistical model and determines the most-likely sequence of subgoals needed to complete the task. The mobile robot, named "Agent Orange," used in these experiments was an ActivMedia Pioneer II DX equipped with a SICK scanning laser range finder, shown in Figure 1. We used the Carnegie Mellon Robot Navigation Toolkit (Carmen) [6] to provide base services such as localization, mapping, and low-level control.

II. Observing the User

The primary sensor used by Dollop is a scanning laser range finder. While this sensor provides extremely accurate measurements, it cannot discriminate between geometrically similar objects in the view plane, such as the waist of a person and a trash can. Therefore Dollop uses extensive processing to extract meaningful information from these measurements. The method for processing laser readings, shown graphically in Figure 4, requires a background occupancy grid map of the

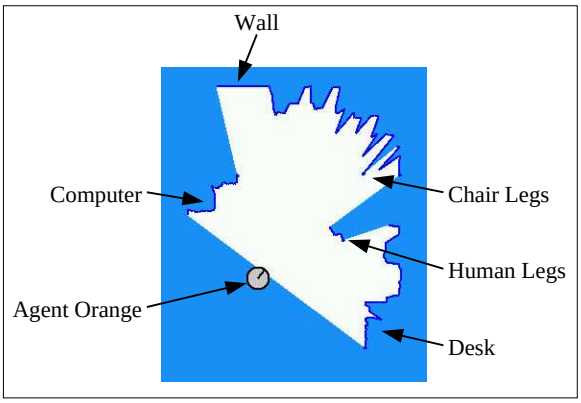


Fig. 3. Sample laser scan.

workspace and is similar to the methodology of Fod et al. [7]. Objects detected by the laser are classified as belonging to one of two classes: background or foreground. Background objects are those that can be explained by the background occupancy grid map, while foreground objects are those that cannot. In these experiments, background objects are things always found in the laboratory: desks, stools, computers, etc. Foreground objects consist of dynamic or static experiment-specific objects, such as humans and slalom cones. A laser reading is designated as a *foreground cell* if there is no background object within some radius, e.g., 20 centimeters, of the measurement. Foreground cells are stored in a separate foreground occupancy grid map, with the cells updated in an autoregressive manner. A foreground cell is considered *occupied* if its value exceeds a predetermined threshold and adjacent occupied foreground cells are called a *blob*. A *foreground object* is a blob that envelops more than a required area, e.g., 100 square centimeters. The position of the foreground object is the centroid of the blob.

We use a Kalman filter [8] to estimate the velocity of a foreground object and to filter out sensor noise, which seems primarily due to the inconsistent nature of computing the centroid of a blob. There are several attractive choices to cope with temporary object occlusion, including particle filters and Kalman predictions. In our experiments, Kalman predictions were more reliable in modeling occluded foreground-object motion. It is well known that estimates of the derivative of a signal amplify noise. We found that incorporating an estimate of velocity (but not acceleration) generally produced the best performance in tracking occluded objects. This implies that our Kalman predictions of occluded objects will proceed in straight lines.

In any practical situation, there will be multiple foreground objects in the workspace at any given time. For tracking purposes, it is necessary to match foreground objects at successive time steps. Dollop uses a greedy search to match the Kalman predictions of foreground objects from time n to the set foreground objects sensed by the laser at time $n+1$. If a Kalman prediction cannot be matched to a sufficiently close foreground object at time $n+1$, then the object is

To appear in Proceedings of the IEEE International Conference on Robotics and Automation
April 26 - May 1, 2004, New Orleans, Louisiana USA

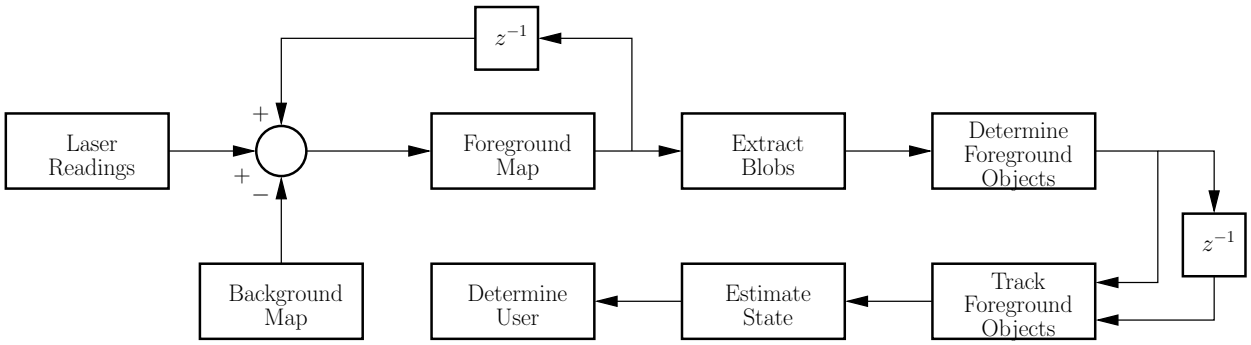


Fig. 4. Flow diagram for observing the user.

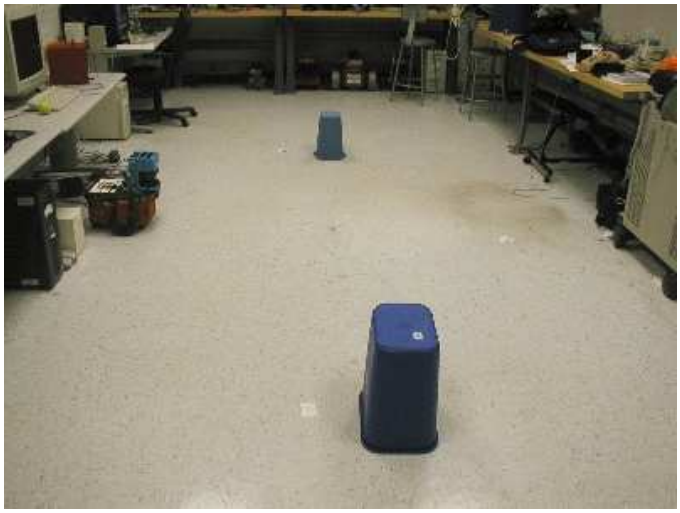
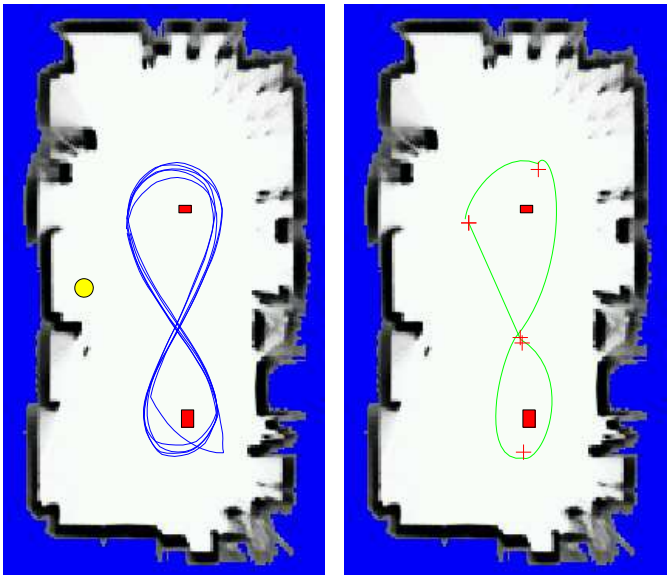


Fig. 5. Example setup of the laboratory with two obstacles in the workspace.



(a) (b)

Fig. 6. Figure 6(a) is the track of the user during five demonstrations of a "figure-eight" trajectory. The red boxes indicate the location of obstacles, and the yellow circle indicates the position of the robot while tracking the user. Figure 6(b) has the subgoals and control law extracted from the demonstrations. The green line represents the control law the robot used to perform the task, based on the sequenced LDS representation. The crosses mark the location of subgoals.

considered *occluded*. If a foreground object remains occluded for more than some amount of time, e.g., 5 seconds, then it is considered *disappeared*. This greedy search for foreground-object tracking works well in environments that are relatively sparse. However, a joint optimization procedure may be more appropriate in workspaces with a large number of coinciding dynamic foreground objects. After matching foreground objects, we then estimate their velocities using Kalman updates. We consider the foreground object with the highest estimated velocity to be the user.

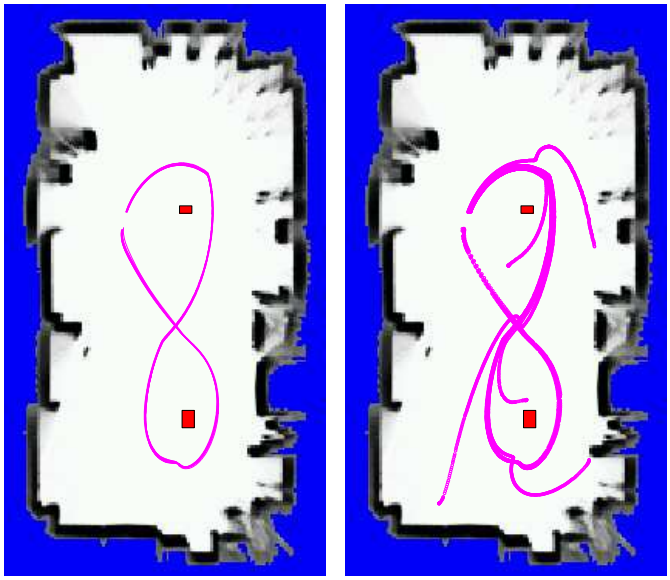
To make these concepts more concrete, we use a background occupancy grid map of a laboratory computed by Carmen. We then placed two obstacles, i.e., trash cans, about four meters apart and asked a user to walk several iterations of a "figure-eight" trajectory while the robot observed from a fixed location, shown in Figure 5. In Figure 6(a), we plot the results of these demonstrations. Though the user is temporarily occluded while demonstrating the trajectory, the filters are able to interpolate the trajectory accurately.[1]

[1]During one demonstration the Kalman filter temporarily lost track of the user. This is manifested by the outlying line on the right side of Figure 6(a).

III. Describing User Trajectories

To reduce the number of parameters used to describe a demonstration, Dollop represents trajectories by a sequence of linear dynamical systems [9]. This method automatically divides a trajectory into a set of segments, with each segment being fitting by a single Linear Dynamical System (LDS). Each LDS is computed from a closed-form least-squares estimate and captures the salient features of the trajectory such as direction, curvature, and speed. The endpoints of each segment specify important points in the trajectory and, in many ways, these points can be considered *subgoals*. This sequenced LDS approach is particularly appealing for LBO applications since it represents trajectories in a *generative* fashion, encoding the trajectory with a stable control law induced by the LDS. That

To appear in Proceedings of the IEEE International Conference on Robotics and Automation
April 26 - May 1, 2004, New Orleans, Louisiana USA



(a) (b)

Fig. 7. Figure 7(a) shows the result of ten runs of Agent Orange performing the estimated trajectory of Figure 6(b). Figure 7(b) shows the response of Agent Orange after being "kidnapped" during five executions of the trajectory.

is, the generative representation provides a mapping from current state to desired state. Since the sequenced LDS approach is based on least-squares estimation, multiple demonstrations of a trajectory can be incorporated to compute a more accurate generalization of the intended user trajectory. Furthermore, the subgoals make modifying the task trivial: a trajectory can be scaled or shifted by simply moving the subgoals and allowing the LDS to interpolate its response. Thus, the system captures the notion of "what the user would do."

In Figure 6(b), we extract a sequence of LDS estimates and subgoals from the trajectories in Figure 6(a). Qualitatively, this estimated trajectory appears to capture the intentions of the original trajectory: a "figure-eight" trajectory that avoids the obstacles. Quantitatively, the estimated trajectory is represented by 6 LDS estimates and has an average error of 20 millimeters compared to the original trajectory.

However, the LDS representation does not respect the dynamics of Agent Orange. The sequenced-LDS representation is optimal for idealized systems, but we make no guarantees for nonholonomic, underactuated, or saturating systems. In Figure 7(a) we plot the response of Agent Orange during ten runs of the estimated task. While it does not exactly follow the idealized trajectory in Figure 6(b), the robot does come fairly close to the predicted path. In Figure 7(b) we plot the response on five runs during which we "kidnap" Agent Orange during execution and plot its response to these perturbed conditions.Because we represent the trajectory in a generative manner, using a supervisory control law induced from the LDS estimates, Dollop automatically interpolates its response after being kidnapped by emultating "what the user would have done." It is coincidental that none of the responses to the kidnapping resulting in Agent Orange hitting the obstacles, or laboratory walls, since the response of the control laws only depends on the current state of the robot and is independent of objects in the environment. However, the response of Agent Orange to being kidnapped seems to be a reasonable estimate of user intent.

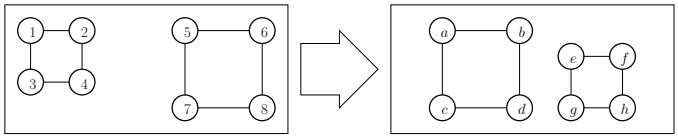


Fig. 8. The eight objects on the left must be matched to those on the right.

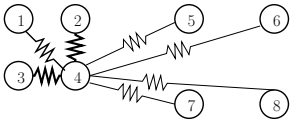


Fig. 9. Intuitive representation of the "spring" optimization.

IV. Environment Configuration

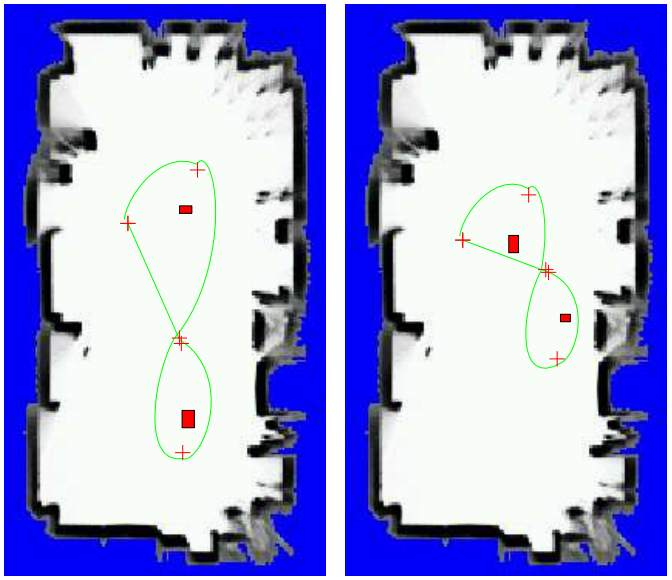
Since demonstrations may be performed in different environments, an LBO system must have the ability to determine how the actions of the user are affected by these changes. To determine the environment configuration, Dollop extracts the set of static foreground objects from the foreground occupancy grid described in Section II, cf. Figure 4. The environment configuration is then defined to be the set of vertices of the bounding boxes enveloping each static foreground object. If there are N static foreground objects, then the environment configuration would contain $4N$ objects, one for each corner of the bounding boxes.

A. Mapping Environment Configurations

A crucial step in learning from, and performing, tasks demonstrated in different environment configurations is determining how different environment configurations relate to each other. For example, in Figure 8 each object on the left, $\{1, 2, 3, 4, 5, 6, 7, 8\}$, must be matched to one on the right, $\{a, b, c, d, e, f, g, h\}$. There are many possible solutions and any objective function giving preference to one possible mapping over another is implicitly assuming that some mappings are more likely than others. We formulate the matching problem by considering each object connected to all others by "springs" with stronger springs connected to closer objects, as in Figure 9. These springs result in a "force" on the object. Matching one object with another results in a change in force and we compute the matching that minimizes the *total* change in force on all matched objects.

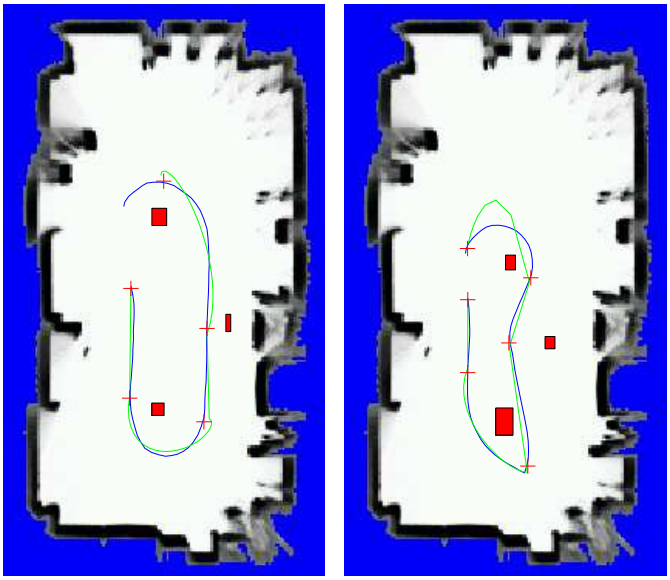
Let the environment configuration of the ith demonstration be given by $\mathbf{\Omega}^i = \{\boldsymbol{\omega}^i_1, \ldots, \boldsymbol{\omega}^i_{4N}\}$ and let the environment configuration of the jth demonstration be given by $\mathbf{\Omega}^j = \{\boldsymbol{\omega}^j_1, \ldots, \boldsymbol{\omega}^j_{4N}\}$. Formally, we are computing the function $g : \mathbf{\Omega}^i \rightarrow \mathbf{\Omega}^j$ that minimizes the cost function $c : \mathbf{\Omega}^i \times \mathbf{\Omega}^j \rightarrow \mathbb{Z}$

To appear in Proceedings of the IEEE International Conference on Robotics and Automation
April 26 - May 1, 2004, New Orleans, Louisiana USA



(a) (b)

Fig. 10. Performing the task from Figure 6(b) in a modified environment. The subgoals have shifted as objects in the environment have changed. The estimated trajectory automatically adapts to "what the user would have done" in Figure 10(b).



(a) (b)

Fig. 11. Two demonstrations of a task in modified environments and the corresponding estimated trajectories.

summed over all matched objects. Let the spring constant between object $\boldsymbol{\omega}_m^i$ and object $\boldsymbol{\omega}_n^i$ be

$$k_{m,n}^i \triangleq \left\|\boldsymbol{\omega}_m^i - \boldsymbol{\omega}_n^i\right\|^{-2}.$$

The total force on object $\boldsymbol{\omega}_m^i$ is

$$\boldsymbol{f}_m^i \triangleq \sum_{\substack{\boldsymbol{\omega}_n^i \in \boldsymbol{\Omega}^i \\ \boldsymbol{\omega}_n^i \neq \boldsymbol{\omega}_m^i}} k_{m,n}^i \frac{\boldsymbol{\omega}_n^i - \boldsymbol{\omega}_m^i}{\left\|\boldsymbol{\omega}_n^i - \boldsymbol{\omega}_m^i\right\|}.$$

If object $\boldsymbol{\omega}_m^i$ is matched with object $\boldsymbol{\omega}_n^j$, then the magnitude change in force caused by this individual match is

$$c(\boldsymbol{\omega}_m^i, \boldsymbol{\omega}_n^j) \triangleq \left\lceil \alpha \left\| \boldsymbol{f}_m^i - \boldsymbol{f}_n^j \right\| \right\rceil,$$

where $\lceil b \rceil$ is the ceiling operation on b, which rounds up $b \in \mathbb{R}$ to the next greatest integer, and α is a large constant, e.g., 10^3, to alleviate numerical rounding. It is straightforward to formulate this as a weighted bipartite-graph matching problem known as the Hungarian Method [10]. This formulation can be solved optimally by linear programming and, in practice, a large number of environment objects can matched in well under a second. The Hungarian Method requires that the mapping, $g(\cdot)$, be a *bijective* function. This means that the number of environment objects must be the same in all demonstrations of a task. In our experiments this has not shown itself as a problem but, in general, this is a restrictive assumption. However, there are many sophisticated, and very complex, matching formulations with more realistic assumptions [11].

B. Associating Subgoals with Environment Objects

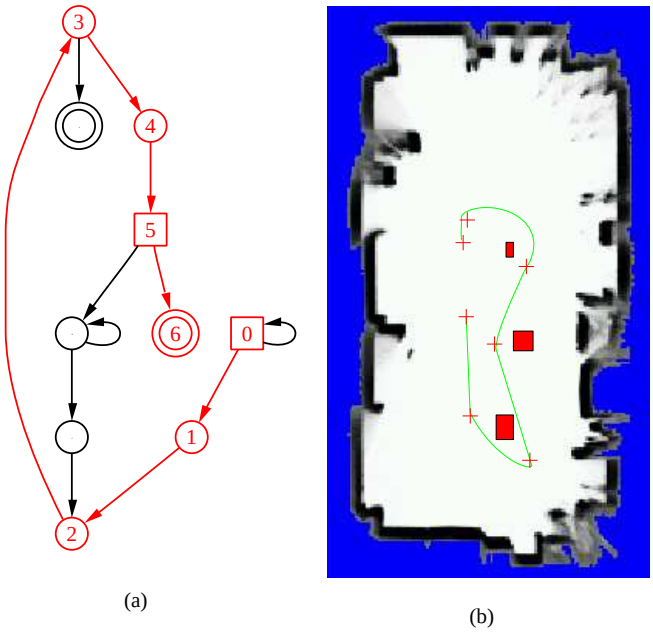
To learn from demonstrations performed in different environments, we compute a hypothesis about what the user would have done if the demonstrations had been performed under the same conditions. This is accomplished by associating subgoals with environment objects, a subject of previous investigations of LBO methods. Morrow and Khosla [12], for instance, extracted the corners of objects in the workspace from camera images. In that work, users were then required to supply the location of the subgoals and manually associate them with the corners. As objects in the workspace moved, a corner-tracking algorithm updated the subgoal locations accordingly. Our method for associating subgoals is conceptually similar but performed automatically. First, Dollop determines the closest m environment objects to a subgoal. The difference between the subgoal and these environment objects is weighted so that closer environment objects have higher weights. These weighted differences are the subgoal associations. Once a mapping between different environments is computed, Dollop updates the location of the subgoals according to how the environment has changed based on the subgoal associations.

In Figure 10, we moved the obstacles from the original "figure-eight" demonstration to different locations in the laboratory. Because the subgoals are associated with the obstacles, their locations change accordingly. Furthermore, each LDS in the estimated trajectory adapts its response to these novel conditions, succeeding in determining what the user would have done: a "figure-eight" trajectory that avoids the obstacles.

V. Learning

We model the user as generating subgoals according to a Continuous-Density Hidden Markov Model (CDHMM). Since we do not know the set of tasks that the user will demonstrate *a priori*, we must estimate the *structure* of the CDHMM from the demonstrations alone. To accomplish this,

To appear in Proceedings of the IEEE International Conference on Robotics and Automation
April 26 - May 1, 2004, New Orleans, Louisiana USA



(a) (b)

Fig. 12. Figure 12(a) shows the CDHMM estimated from the demonstrations in Figure 11, with the most-likely path shown in red. Figure 12(b) shows the result after mapping the subgoals to the new environment and automatically adapting the response of the LDS estimates.

we use an algorithm originally created to predict waypoints in manipulator-robot programs [13]. When applied to Dollop, this algorithm analyzes the sequences of subgoals from different demonstrations of a task. If the algorithm finds statistical similarities between the subgoals of different demonstrations, then the CDHMM is simplified by combining the similarities. This merging process gives an increasingly accurate estimate about where the user intended the subgoals to be located. Furthermore, when segments of the task are merged together, Dollop estimates a single LDS that describes *both* segments, which improves the generalization ability of the estimated trajectory [9]. Both of these attributes mean that Dollop is able to reproduce *and* generalize user actions if it has access to multiple demonstrations performed under different environment configurations, essentially, giving Dollop a better idea of how the user would perform in varying conditions.

For example, in Figure 11 we show two demonstrations of a task performed in different environments, as well as the subgoals and trajectories estimated by Dollop. Before learning occurs, the subgoals from both demonstrations are mapped to the same environment configuration. Dollop then estimates a CDHMM that describes the demonstrations and computes the most-likely sequence of subgoals needed to complete the task, as in Figure 12(a). When faced with a new environment configuration, Dollop maps this sequence of subgoals to the new environment configuration and allows the LDS estimates to adapt their response accordingly, as shown in Figure 12(b).

VI. Conclusions and Future Work

In this paper we presented a system that pursues a computational approach to Learning By Observation, called Dollop. Our approach allows Dollop to learn motor-skill tasks by observing user demonstrations with its scanning laser range finder. From these demonstrations, subgoals are extracted and associated with objects in the environment so that as these objects move, the subgoals move accordingly. We described how Dollop incorporates multiple demonstrations in different environments to extract the intentions of the user. We also showed examples of Dollop learning from demonstrations in a laboratory setting with a mobile robot.

The representation of the environment is probably the weakest point in Dollop. While we can compute environment matchings quickly and consistently, it is an inherently ill-posed problem. The geometric information obtained from laser reading disregards many cues that could be helpful in disambiguating the matching process. As such, it is not difficult to contrive environments that result in counter-intuitive matchings. Coupling laser readings with a more expressive sensor, such as a camera, could result in improved performance. In the future, we plan on extending Dollop to encode higher-level knowledge, beyond motor-skill learning, while still retaining its computational representation.

Acknowledgment

We would like to thank the Intel Corporation for providing the computing hardware. This work was sponsored by ABB Corporate Research.

References

[1] S. Iba, C. J. Paredis, and P. K. Khosla, "Intention aware interactive multi-modal robot programming," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2003.

[2] D. Pomerleau, "Efficient training of artificial neural networks for autonomous navigation," *Neural Copmutation*, vol. 3, no. 1, 1991.

[3] J. Chen and A. Zelinsky, "Programming by demonstration: Coping with suboptimal teaching actions," *International Journal of Robotics Research*, vol. 22, no. 5, 2003.

[4] M. N. Nicolescu, "A framework for learning from demonstration, generalization and practice in human-robot domains," Ph.D. dissertation, Department of Computer Science, University of Southern California, 2003.

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

[6] M. Montemerlo, N. Roy, and S. Thrun, "Perspectives on standardization in mobile robot programming : The Carnegie Mellon navigation (CARMEN) toolkit," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2003.

[7] A. Fod, A. Howard, and M. J. Matarić, "A laser-based people tracker," in *Proceedings of the IEEE International Conference on Robotics and Automation*, 2002.

[8] R. F. Stengel, *Stochastic Optimal Control: Theory and Applications*. Wiley-Interscience, 1986.

[9] K. R. Dixon and P. K. Khosla, "Trajectory representation using sequenced linear dynamical systems," in *Proceedings of the IEEE International Conference on Robotics and Automation*, 2004.

[10] C. H. Papadimitriou and K. Steiglitz, *Combinatorial Optimization: Algorithms and Complexity*, 2nd ed. Dover Publications, 1998.

[11] J. Cheriyan, "Randomized $\tilde{O}(M(|V|))$ algorithms for problems in matching theory," *SIAM Journal on Computing*, vol. 26, no. 6, 1997.

[12] J. D. Morrow and P. K. Khosla, "Sensorimotor primitives for robotic assembly skills," in *Proceedings of the IEEE International Conference on Robotics and Automation*, 1995.

[13] K. R. Dixon, J. M. Dolan, and P. K. Khosla, "Predictive robot programming: Theoretical and experimental analysis," *To appear in International Journal of Robotics Research*, 2004.