Learning Equivalent Action Choices from Demonstration

Sonia Chernova and Manuela Veloso
Computer Science Department
Carnegie Mellon University
Pittsburgh, PA, USA
{soniac,veloso}@cs.cmu.edu

Abstract— **In their interactions with the world, robots inevitably face situations in which multiple actions are equivalently applicable. These situations violate the common assumption that for any world state there exists a single best action. When learning from demonstration, this ambiguity frequently results in inconsistent demonstrations from the teacher, however, the problem of action choices has been overlooked by previous approaches for demonstration learning. In this paper, we present an algorithm that identifies regions of the state space with conflicting demonstrations and enables the choice between multiple actions to be represented explicitly within the robot's policy. An experimental evaluation of the algorithm in a real-world obstacle avoidance domain shows that reasoning about action choices significantly improves the robot's learning performance.**

I. Introduction

In their interaction with the world, robots must constantly make decisions about what action to perform. Policies control the decision making process, providing a mechanism for calculating which action must be taken given a particular world state. *Teaching by demonstration* is a method for interactively training robot policies through demonstrations of the desired robot behavior by a human teacher.

Operating in rich environments, robots inevitably encounter situations in which multiple actions are equivalently applicable. For example, a moving robot that encounters an obstacle directly in its path has the option of moving left or right to avoid it. If the space is empty, both directions are equally valid for performing the desired task. Similar choices can arise in many other situations, such as deciding among objects of equal value. Faced with a choice of equivalent actions, a human teacher often does not select the same action consistently for demonstration. As a result, training data obtained by the robot lacks consistency, with identical, or nearly identical, states becoming associated with different actions.

Existing demonstration learning algorithms have overlooked the problem of action choices, making the common assumption that for any world state there exists a single best action [3], [5], [7], [11]. While it is possible to achieve a non-deterministic state-action mapping with these algorithms, special actions must be defined that abstract this choice. For example, the action `Random` can be used to represent the random selection among all available actions. The appropriate use of `Random` must then be demonstrated by the teacher in order to be incorporated into the robot's policy. While this approach achieves the desired randomized behavior, the algorithm does not explicitly represent the existence of multiple equivalently applicable actions.

In this paper, we enable the robot to reason about action choices. We build upon two existing algorithms, Confident Execution [5], [6] and Corrective Demonstration [4], which enable us to obtain task demonstrations from the teacher and learn the robot policy. Based on this information, we present an algorithm for identifying regions of the state space in which data from multiple classes overlaps as a result of inconsistent demonstrations. For these regions, we make the assumption that the robot's action can be selected at random among the conflicting data classes. The choice between multiple actions is then modeled explicitly within the robot's action policy through *option classes*. This automated approach does not require the teacher to predefine or demonstrate special choice actions, extending instead from the person's natural demonstration technique.

In the following section we discuss related work in the area of learning from demonstration. Section III presents a description of the obstacle avoidance domain used throughout this paper. Section IV presents the base demonstration learning algorithm consisting of Confident Execution and Corrective Demonstration and discusses its limitations with respect to dealing with multiple equivalent actions. Section V then presents our solution to the problem of action choices, followed by experimental results and evaluation in Section VI.

II. Related Work

A wide variety of approaches for teaching robots through demonstration have been proposed in recent work. Lockerd and Breazeal [3], [8] present a robotic system in which high-level tasks are taught through social interaction. Speech and visual queues are used to guide the robot and perform demonstrations, while the robot additionally expresses its internal state through emotive cues, such as facial and body expressions, to help guide the teaching process. Based on these demonstration, a high level hierarchical task model is learned using the Bayesian likelihood method.

Nicolescu and Mataric [10], [11] introduce a plan-based framework in which training is performed by having the robot follow a human and observe its actions. A high level task representation is constructed by the algorithm by analyzing this experience with respect to the robot's abilities.

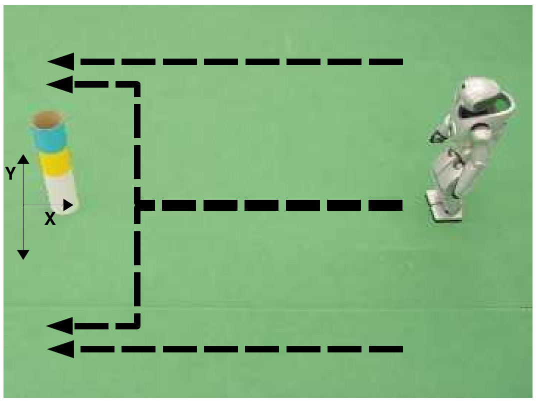


Fig. 1. Obstacle avoidance domain with Sony QRIO robot.

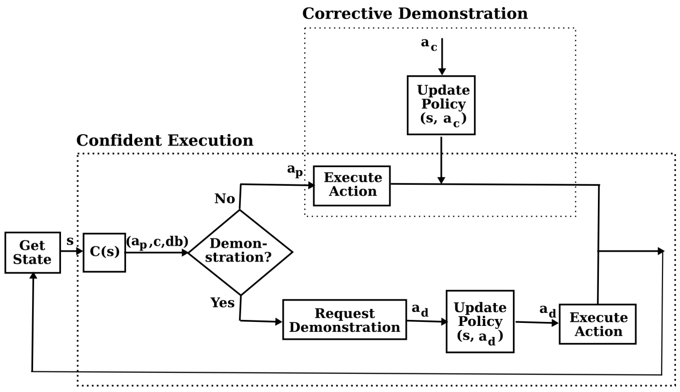


Fig. 2. Diagram of the combined Confident Execution and Corrective Demonstration learning process.

Several approaches utilizing classification and regression for policy learning from demonstration have also been proposed. Bentivegna et al. [1], [2] and Saunders et al. [12] use the k-nearest neighbor (KNN) [9] algorithm to classify instances based on similarity to training examples, resulting in a policy mapping from sensory observations to actions. Grollman and Jenkins apply Dogged Learning, a confidence-based learning approach based on Locally Weighted Projection Regression, to teaching low level robotic skills such as ball seeking and grabbing [7]. All of these approaches, however, assume that for each state there exists a single best action.

III. Domain

A robot obstacle avoidance domain is used to compare and evaluate learning algorithms throughout this paper. In this domain, shown in Figure 1, a Sony QRIO humanoid robot is taught to navigate around a colored post located in its path. The robot has a choice of three actions, Forward (F), Left (L) and Right (R), each of which moves it approximately 10cm in the designated direction. The robot's state is represented by its relative position to the post in two-dimensional space (x,y). A noisy estimate of the post's position is obtained from the robot's onboard vision system.

The lines in Figure 1 represent example robot trajectories, with each dash approximately equal to the length of a single robot step. Note that when the robot is positioned directly in front of the post, both avoidance directions, left and right, are equivalently applicable. Once the robot has committed to a direction and has shifted its position, however, it must commit to this course to avoid oscillation. When faced with a choice of directions during demonstration, the teacher selects among the alternate actions at random.

IV. Demonstration Learning Algorithm

In this section, we present a summary of the Confident Execution [5], [6] and Corrective Demonstration [4] algorithms used to obtain teacher demonstrations and learn the action policy. Figure 2 presents an overview of the complete learning process. In Section IV-C we discuss how the assumption of one-to-one state-action mapping made during policy learning presents serious limitations in domains with multiple applicable actions.

A. *Confident Execution*

Confident Execution is an interactive learning algorithm in which the robot must select demonstration examples, in real time, as it interacts with the environment. At each timestep, the algorithm obtains the current robot state and determines whether a demonstration of the correct action in this state will provide useful information and improve the robot's policy. If demonstration is required, the robot requests help from the teacher and waits for the person to select an action. Upon obtaining the demonstration, the algorithm updates the robot's policy using the acquired action label and performs the demonstrated behavior. If a demonstration is not required, the robot autonomously executes the action specified by its policy without consulting the teacher.

The robot's policy is represented and learned using supervised learning based on training data acquired from the demonstrations. The policy is represented by classifier $\mathcal{C} : s \rightarrow (a, c, db)$, trained using state vectors s_i as inputs, and actions a_i as labels. For each classification query, the model returns the policy action $a \in \mathcal{A}$, action selection confidence c, and the decision boundary db with the highest confidence for the query. In this work, the policy is represented by a set of Gaussian Mixture Models (GMMs) with individual Gaussian components forming the decision boundaries.

Given a new state, the algorithm selects between demonstration and autonomy based on the measure of action selection confidence. Specifically, the state is classified, and autonomous execution is selected if the value of classification confidence c is above a threshold. The algorithm maintains a set of thresholds, one per decision boundary, that are customized to the underlying data distribution. The threshold corresponding to the highest confidence decision boundary, db, is used to determine autonomy for a particular query.

The confidence threshold for each decision boundary is calculated based on the confidence scores of datapoints misclassified by that boundary. For example, consider a Gaussian component that mistakenly classifies a number of points from a different data class. For this component, the threshold is set to the average classification confidence of the misclassified points. This technique is based on the assumption that "future classifications with confidences at or below this value are likely to be misclassifications as well". Using this technique, individual thresholds act to isolate regions of the state space in which classification is unreliable. Future robot states that fall within this region of uncertainty trigger additional demonstration requests from the teacher with the aim of obtain clarity through additional data. Learning is complete once the robot is repeatedly able to perform the correct behavior without requesting additional demonstrations. Full details of the Confidence Execution learning and threshold calculation algorithms can be found in [5], [6].

B. *Corrective Demonstration*

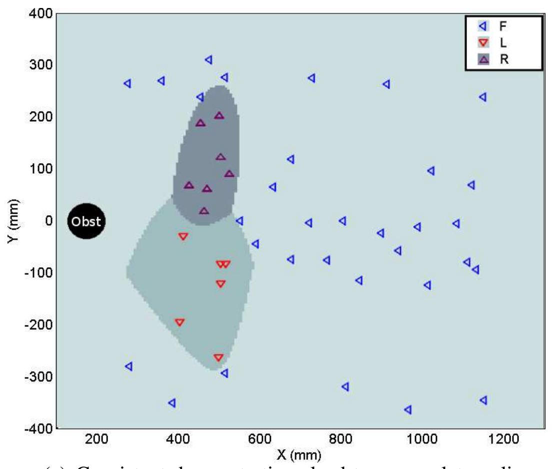
The above Confident Execution algorithm enables the robot to identify unfamiliar and ambiguous states and prevents autonomous execution in these situations. However, states in which an incorrect action is selected with high confidence can still occur, typically due to over-generalization of the classifier. When allowing the robot to select demonstration and regulate its own autonomy, it is important to provide a mechanism for correcting unwanted behavior. The Corrective Demonstration algorithm enables the teacher to perform additional demonstrations and retroactively correct execution mistakes.

As illustrated in Figure 2, Corrective Demonstration comes into play each time the robot executes an autonomous action. During the execution of an autonomously selected action, the teacher has the option to perform a corrective demonstration. This demonstration indicates to the robot that the wrong action was selected, and specifies which action should have been performed in its place. A training point consisting of the original decision state coupled with the corrective action is then used to update the robot's policy. This approach gives the teacher greater control over the learning process and enables the robot to learn quickly from its mistakes.

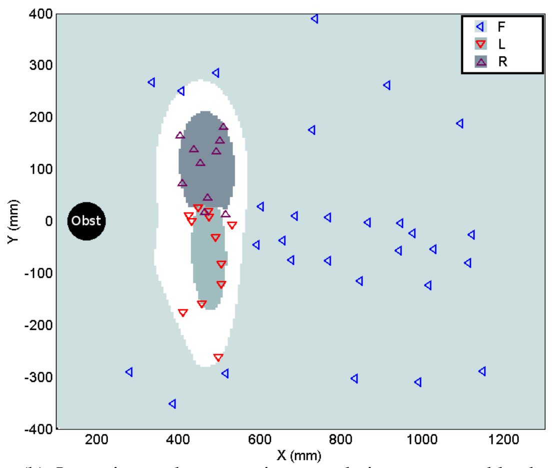
C. *Algorithm Limitations*

The Confident Execution policy learning algorithm has been shown to work effectively, enabling the robot to learn from a small number of demonstrations. However, the algorithm makes two strong assumptions: 1) that for each state there exists a single best action, and 2) that the teacher is able to demonstrate this state-action combination consistently. Based on these assumptions, the algorithm further assumes that a *complete* policy, one that classifies the entire state space with high confidence and results in full robot autonomy (no misclassified points and confidence thresholds of 0), can be achieved if enough demonstrations are obtained.

However, these assumptions frequently fail for robots operating in real-world environments. As described earlier, one common reason is action ambiguity due to the existence of multiple equivalent actions. Consider the obstacle domain, for example, in which in addition to the direction of movement, the teacher must also decide the distance from the post



(a) Consistent demonstrations lead to a complete policy.



(b) Inconsistent demonstrations result in non-separable data classes and an incomplete policy with low confidence regions (white).

Fig. 3. Policies in the obstacle avoidance domain.

at which to begin to avoid the obstacle. Should avoidance actions be initiated at a distance of 1 meter, 0.5 meters, or somewhere in between? In reality, a whole range of distances are likely to provide the desired behavior, and over this range multiple actions are equally valid (move forward a little more, move left or move right). The result is that the teacher's demonstrations are likely to be inconsistent.

Another common reason for inconsistent demonstration is robot sensor noise. Consider, for example, an obstacle avoidance domain in which the teacher closely observes the robot's behavior and consistently selects the same actions to perform. If these demonstrations are selected based on the person's observation of the robot's position, they will not take into account possible noise and bias present in the robot's sensors. As a result, these demonstrations may still appear to be inconsistent to the robot.

In summary, despite the teacher's best intentions, inconsistent demonstrations are extremely likely to occur. An example in Figure 3 shows the effect inconsistent demonstrations have one the Confident Execution policy. Images in this figure show two example distributions obtained from demonstration learning in the obstacle avoidance domain. Each point in the figure represents a demonstration performed when the robot was at that (x,y) position. Background shading represents the most likely action class, with white representing low-confidence (demonstration) regions.

In Figure 3(a), consistent demonstrations result in a complete, fully autonomous policy with no low-confidence regions. In Figure 3(b), inconsistent demonstrations in the middle region of the graph result in non-separable, overlapping datapoints and an incomplete policy. As the robot repeats the task, all states that fall within the low confidence, white, region trigger additional demonstrations. However, unlike the consistent demonstration case, acquiring additional data points will not help the classifier. Instead, an infinite number of overlapping demonstration points will be collected by the algorithm. In the following section we present our algorithm for identifying such regions of the state space and modeling them explicitly as multi-action classes.

V. Option Classes

To address the problem of inconsistent demonstrations stemming from the existence of multiple equivalent actions or sensor noise we introduce *option classes*, which we define as data classes that represent the choice between two or more actions. In this section we present an algorithm for extracting option classes from the underlying data distribution, enabling action choices to be modeled explicitly by the robot's policy.

Option classes represent the choice between any number of available actions. For the obstacle avoidance domain, example option classes include Option-Forward-Left ($O\text{-}FL$), Option-Left-Right ($O\text{-}LR$), and Option-Forward-Left-Right ($O\text{-}FLR$) in which any of the robot's three available actions are applicable. Figure 4 presents an example of the option class algorithm applied to the distribution from Figure 3(b). Note that the middle region of overlapping points now forms the option class $O\text{-}RL$, representing the choice between two valid actions, Right and Left. With the addition of this new data class the task can be represented by a complete policy.

The goal of our algorithm is to identify regions of the state space in which data from multiple classes overlaps and is not separable. For any such region, we make the assumption that the robot's action can be selected at random among the action classes represented by these datapoints. The randomized selection is biased based on the distribution of the action classes, so that more frequently demonstrated actions are more likely to be selected.

Algorithm 1 presents the option class algorithm, which is executed each time the classifier is retrained following a demonstration. Given the complete set of demonstrations D, the algorithm identifies the set of datapoints M that fall within the low confidence region and can not be classified with high confidence (line 2). Additionally, the algorithm calculates the average nearest-neighbor distance over the complete demonstration dataset (line 3). Nearest neighbor distance is defined as the Euclidean distance from the query point to the closest point in the data set. The average nearest

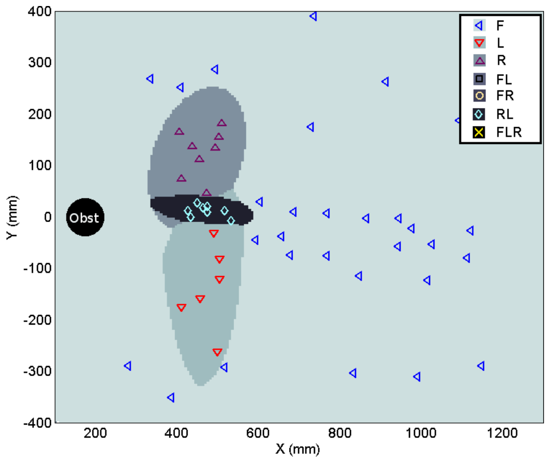


Fig. 4. Option class policy obtained from the inconsistent demonstration distribution in Figure 3(b).

Algorithm 1 Option Class Algorithm

```
1: given demonstration dataset $D$
2: $M \leftarrow$ PointsInLowConfidenceRegion($D$)
3: $d \leftarrow$ MeanNearestNeighborDist($D$)
4: $C \leftarrow$ ConnectedComponents($M$,$d$)
5: for $c \in C$ do
6:     $A \leftarrow$ ActionClasses($c$)
7:     if Size($c$) $> 3$ and Size($A$) $> 1$ then
8:         CreateClass($D$, $c$, $Option\text{-}A$)
9: UpdateClassifier($D$)
10: ResetClass($D$)
```

neighbor distance over the complete dataset provides the algorithm with a domain-independent method for evaluating point proximity and identifying points that are located close together.

Dataset M and mean nearest neighbor distance d are used to identify clusters of closely located misclassified points. Searching over set M, the algorithm identifies points that form connected components with a maximum distance d between points (line 4). The resulting set C contains clusters of points that form the candidate option classes. For each set of points forming a connected component, the algorithm calculates the action classes A represented by the datapoints.

If the cluster contains more than three points and consists of two or more action classes, points from the connected component are used to form a new option class representing the choice between the actions in A (lines 8). Action labels for these points are temporarily changed from their original demonstrated action to $Option\text{-}A$. Once all connected components are analyzed, the classifier representing the robot's policy is relearned with the new option class labels.

Once a policy representing the new option classes has been obtained, action labels for all datapoints in dataset D are reset to their original demonstrated action. The option class acts only as a temporary label and must be re-acquired at the next policy update. This mechanism is used to ensure that any unnecessary option classes that form early in the learning process due to insufficient demonstrations are dissolved once additional demonstrations are obtained.

In summary, while option classes enable the robot's policy to explicitly model action choices, they remain an abstract internal representation within the algorithm. Actions representing option classes are not made available to the teacher for demonstration. As a result, the algorithm never acquires training points with option class labels, but must learn them based on the underlying distribution of other action classes.

Additionally, we define special rules for option classes with regards to classification. Based on the assumption that option classes represent a choice between several valid actions, we specify that a datapoint with label x can not be misclassified by an option class representing the choice between actions in A if $x \in A$. For example, points that belong to class F or L and fall into class $O\text{-}FL$ are not considered to be misclassified. The opposite case also holds, such that points in $O\text{-}FL$ are not misclassified if they fall into F or L. This rule helps to eliminate low confidence regions that may be caused by such misclassifications.

The contributed option class algorithm is classifier independent and contains only a single variable, d, the average nearest neighbor distance. This variable is domain independent because its value is calculated by the algorithm based on the data distribution specific to the applied domain. However, the behavior of the algorithm can be controlled by scaling the value of d. Smaller values will require dense clusters of points to form option classes, requiring a large number of demonstrations. Larger values will lead to greater generalization and the formation of large option classes from relatively few points. Through experimental evaluation we have found that scaling the mean nearest neighbor distance value by 1.5 provides a good balance between generalization and the number of demonstrations.

VI. Experimental Results

In this section, we compare the performance of the Confident Execution algorithm with and without option classes in the obstacle avoidance domain. Corrective Demonstration was applied for both algorithms as necessary.

For the evaluation, we recorded 50 complete demonstration trials of the obstacle avoidance task. Data from the trials, shown in Figure 5, was used for algorithm comparison, providing a consistent library of training points. For each evaluation, demonstration trials from the library were presented to the learning algorithm in random order.

The performance of each algorithm was compared with respect to the number of demonstrations and the percentage of complete policies at the end of the training sequence. A count of the number of demonstrations requested by the robot through Confident Execution (CE) and selected by the teacher through Corrective Demonstrations (CD) was also maintained, along with the number of demonstrations falling within the circled option regions (OR) highlighted in Figure 5. Since multiple valid actions exist for all option regions, classification accuracy is not an informative metric for this evaluation. The following table presents the results, averaged over 20 trials for each algorithm:

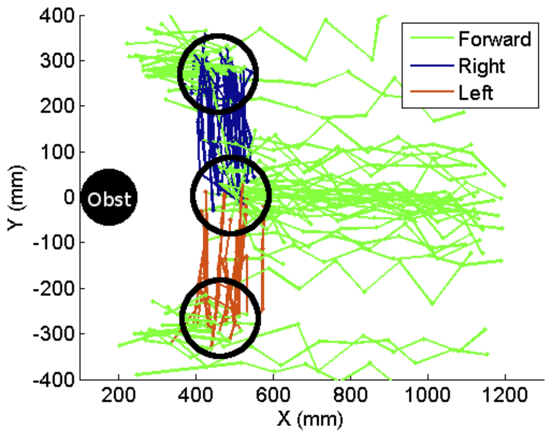


Fig. 5. Points from 50 demonstration trials used for algorithm evaluation and comparison. Circles highlight areas of inconsistent demonstration.

	Demonstrations				Com-
	Total	CE	CD	OR	plete
Conf. Exec.	99.8	95.7	4.1	42.8	20%
Conf. Exec. + Opt.	67.2	55.9	11.3	15.1	90%

The results show a significant improvement in learning performance due to option classes. Given the same library of training points, the original Confident Execution algorithm resulted in a complete action policy in only 20% of trials. The larger number of demonstration requests seen with this algorithm is attributed largely to the remaining 80% of trials in which non-separable points formed low confidence regions in which large numbers of demonstrations were requested. The difference in the number of demonstrations performed within the option regions accounts for most of the difference in the total number of demonstrations between algorithms.

Using option classes, the Confident Execution algorithm resulted in a complete action policy in 90% of experimental trials, requiring an average of 67.2 demonstrations to cover the state space. Figure 6 presents three example policies resulting from the experiments. Each policy shows a different, but equally valid combination of option classes.

Figure 6(a) presents a policy consisting of four option classes. A small region centered near (520,0) represents the option class $O\text{-}FRL$, in which any of the three robot actions are valid. As the robot nears the obstacle, the option class becomes $O\text{-}RL$, indicating that the Forward action is no longer a valid option. Option classes $O\text{-}FR$ and $O\text{-}FL$ represent the other option regions. In Figure 6(b) only three option classes form, with a large $O\text{-}RL$ as the robot approaches the obstacle. In Figure 6(c), points within the middle option region are fully separable, resulting in a policy with a slight bias towards passing on the right and only a single option class, $O\text{-}FL$. Each of these policies was successfully applied to performing the obstacle avoidance task using the real QRIO robot.

VII. Conclusion

Regions of the state space in which consistent demonstration is impossible due to the existence of multiple applicable actions or noise are frequently encountered in learning from demonstration. Existing demonstration learning algorithms have overlooked this problem, making the assumption that for any state the robot must learn the single best action to perform. In this paper, we introduced a classifier-independent algorithm that enables choices between multiple actions to be represented explicitly within the robot's policy through the creation of option classes. Evaluation and comparison of demonstration learning with and without option classes was performed in a real-world, noisy robot domain. Option classes were shown to significantly improve learning performance, enabling the algorithm to converge to a complete policy with far greater frequency and requiring fewer demonstrations.

References

[1] D. C. Bentivegna, C. G. Atkeson, and G. Cheng. Learning from observation and practice using primitives. *AAAI Fall Symposium Series, 'Symposium on Real-life Reinforcement Learning'*, 2004.

[2] D. C. Bentivegna, A. Ude, C. G. Atkeson, and G. Cheng. Learning to act from observation and practice. *International Journal of Humanoid Robotics*, 1(4), 2004.

[3] C. Breazeal, G. Hoffman, and A. Lockerd. Teaching and working with robots as a collaboration. In *3rd Int. Joint Conference on Autonomous Agents and Multiagent Systems*, pages 1030–1037, Washington, DC, USA, 2004. IEEE Computer Society.

[4] S. Chernova and M. Veloso. Teaching multi-robot coordination using demonstration. In *Under submission.*

[5] S. Chernova and M. Veloso. Confidence-based policy learning from demonstration using gaussian mixture models. In *Proceedings of Int. Joint Conference on Autonomous Agents and Multiagent Systems*, May 2007.

[6] S. Chernova and M. Veloso. Multi-thresholded approach to demonstration selection for interactive robot learning. In *Proceedings of 3rd ACM/IEEE International Conference on Human-Robot Interaction (HRI'08)*, March 2008.

[7] D. Grollman and O. Jenkins. Dogged learning for robots. In *IEEE International Conference on Robotics and Automation*, pages 2483–2488, 2007.

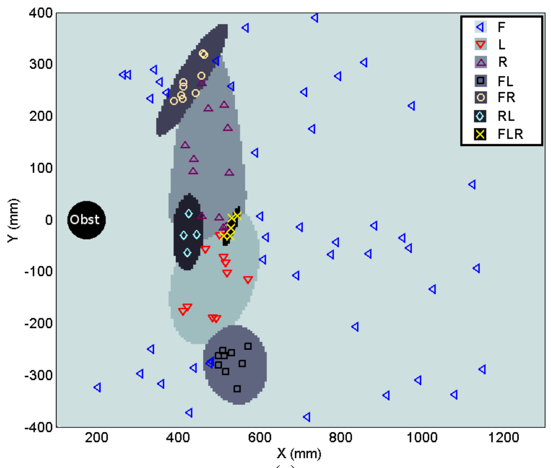
[8] A. Lockerd and C. Breazeal. Tutelage and socially guided robot learning. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2004.

[9] T. Mitchell. *Machine Learning*. McGraw Hill, 1997.

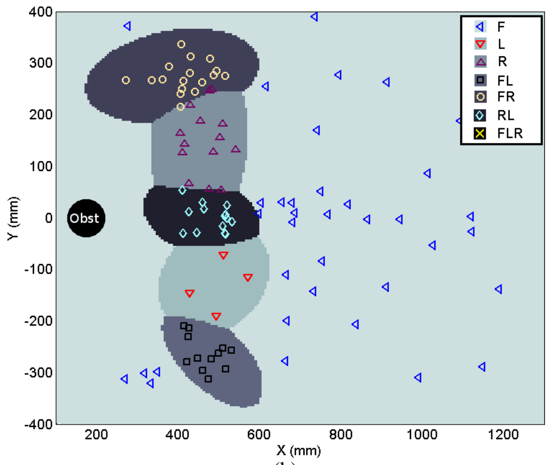
[10] M. N. Nicolescu and M. J. Mataric. Learning and interacting in human-robot domains. In *IEEE Transaction on Systems, Man and Cybernetics*, pages 419–430, 2001.

[11] M. N. Nicolescu and M. J. Mataric. Natural methods for robot task learning: instructive demonstrations, generalization and practice. In *Proceedings of AAMAS'03*, pages 241–248. ACM Press, 2003.

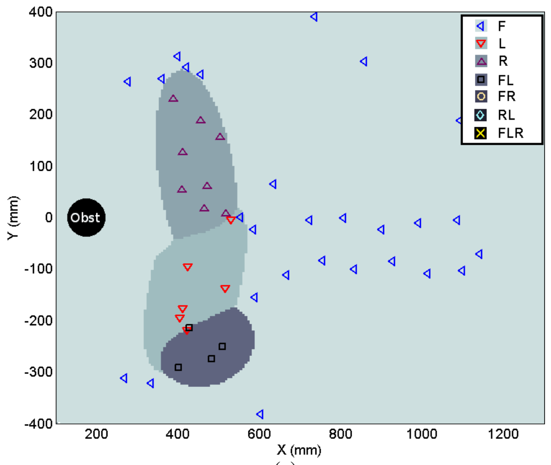
[12] J. Saunders, C. L. Nehaniv, and K. Dautenhahn. Teaching robots by moulding behavior and scaffolding the environment. In *Proceeding of the 1st Conference on Human-Robot Interaction*, pages 118–125, New York, NY, USA, 2006. ACM Press.



(a)



(b)



(c)

Fig. 6. Three option class policies for the obstacle avoidance domain.