# Confidence-Based Policy Learning from Demonstration Using Gaussian Mixture Models

Sonia Chernova
Carnegie Mellon University
Computer Science Department
Pittsburgh, PA, USA
soniac@cs.cmu.edu

Manuela Veloso
Carnegie Mellon University
Computer Science Department
Pittsburgh, PA, USA
veloso@cs.cmu.edu

## ABSTRACT

We contribute an approach for interactive policy learning through expert demonstration that allows an agent to actively request and effectively represent demonstration examples. In order to address the inherent uncertainty of human demonstration, we represent the policy as *a set of Gaussian mixture models* (GMMs), where each model, with multiple Gaussian components, corresponds to a single action. Incrementally received demonstration examples are used as training data for the GMM set. We then introduce our *confident execution* approach, which focuses learning on relevant parts of the domain by enabling the agent to identify the need for and request demonstrations for specific parts of the state space. The agent selects between demonstration and autonomous execution based on statistical analysis of the uncertainty of the learned Gaussian mixture set. As it achieves proficiency at its task and gains confidence in its actions, the agent operates with increasing autonomy, eliminating the need for unnecessary demonstrations of already acquired behavior, and reducing both the training time and the demonstration workload of the expert. We validate our approach with experiments in simulated and real robot domains.

## Categories and Subject Descriptors

I.2.6 [**Artificial Intelligence**]: Learning—*Machine Learning*

## General Terms

Algorithms, Design, Performance

## Keywords

learning from demonstration, imitation, robotics

## 1. INTRODUCTION

Designing robot controllers is a challenging problem due to sensor complexity, noise, and the non-deterministic effects of robot actions and the environment. As a result, many robotic platforms are not used to their full potential due to the difficulty of developing controllers for a wide range of applications. To advance robotics and incorporate robots into our daily lives, natural and intuitive approaches must be developed that allow new skills to be taught in a timely manner.

Learning from demonstration, a collaborative learning approach based on human-robot interaction, offers a promising solution to this problem. The goal of this approach is to learn to imitate the behavior of a teacher by observing a demonstration of the task. In addition to providing an intuitive training method, it has been shown to significantly reduce learning time compared to exploration-based methods such as reinforcement learning [22].

We view learning from demonstration as strongly related to statistical supervised learning since both approaches rely on labeled training data. Supervised learning techniques are frequently used in demonstration-based learning to learn a policy given a fixed set of labeled data [4, 10, 18]. In addition to this, demonstration learning must also provide a method for incrementally gathering this training data, ideally in a way that minimizes the number of demonstrations needed to acquire the policy.

In this work, we contribute an interactive policy learning approach that reduces the number of required demonstrations by allowing an agent to actively request relevant demonstration examples. In order to address the inherent uncertainty of human demonstration, we represent the policy as *a set of Gaussian mixture models*, where each model, with multiple Gaussian components, corresponds to a single action. We then introduce our *confident execution* approach, which focuses learning on relevant parts of the domain by enabling the agent to identify the need for and request demonstrations for specific parts of the state space.

Our approach relies on expert demonstration, during which the agent is fully under the control of an expert while continuing to experience the task through its own sensors. Demonstration data is acquired incrementally though confident execution, during which the agent actively decides between autonomously executing the next action and requesting demonstration from the expert based on statistical analysis of the uncertainty of the currently learned GMM set. The agent operates with increasing autonomy as it achieves proficiency at its task, eliminating the need for unnecessary demonstrations of already acquired behavior, and reducing both the training time and the demonstration workload of the expert. Additionally, the quality of the learned model is continually evaluated by comparing its similarity to the demonstrated behavior, allowing the expert to track the agent's learning progress in real time. We validate our approach with experiments in simulated and real robot domains.

## 2. RELATED WORK

Learning from demonstration, and related areas such as learning from observation, imitation, and robot shaping, are interactive learning methods that utilize expert examples. A wide variety of demonstration training methods have been explored in previous work, including teleoperation [18, 25], direct manipulation of the learning agent [3], and following robotic [9] or human [14, 15] teachers. In our work we focus on a direct policy learning approach [20], in which the agent experiences the demonstration through its sensors while under the control of an expert.

Nicolescu and Mataric [14, 15] present a learning framework based on demonstration, generalization and teacher feedback, in which training is performed by having the robot follow a human and observe its actions. A high-level task representation is then constructed by analyzing the experience with respect to the robot's underlying capabilities. The authors also describe a generalization of the framework that allows the robot to interactively request help from a human in order to resolve problems and unexpected situations. This interaction is implicit as the agent has no direct method of communication; instead, it attempts to convey its intentions by communicating though its actions.

Lockerd and Breazeal [7, 11] demonstrate a robotic system where high-level tasks are taught through social interaction. In this framework, the teacher interacts with the agent through speech and visual inputs, and the learning agent expresses its internal state through emotive cues such as facial and body expressions to help guide the teaching process. The outcome of the learning is a goal-oriented hierarchical task model. In later work [23], the authors examine ways in which people give feedback when engaged in an interactive teaching task. Although the study's focus is to examine the use of a human-controlled reward signal in reinforcement learning, the authors also find that users express a desire to guide or control the agent while teaching. This result supports our belief that, for many robotic domains, teleoperation provides an easy and intuitive human-robot communication method.

Bentivegna et al. [4, 5, 6] and Saunders et al. [18] present demonstration learning approaches based on supervised learning methods. Both groups use the $k$-nearest neighbor (KNN) [12] algorithm to classify instances based on similarity to training examples, resulting in a policy mapping from sensory observations to actions.

Our approach similarly uses supervised learning, in the form of Gaussian mixture models, for classification. Additionally, we contribute an interactive learning approach that allows the agent to request help from a human, similar to that of Nicolescu and Mataric [14, 15]. Inamura et al. [10] present a similar method based on Bayesian Networks [16] limited to a discretely-valued feature set.

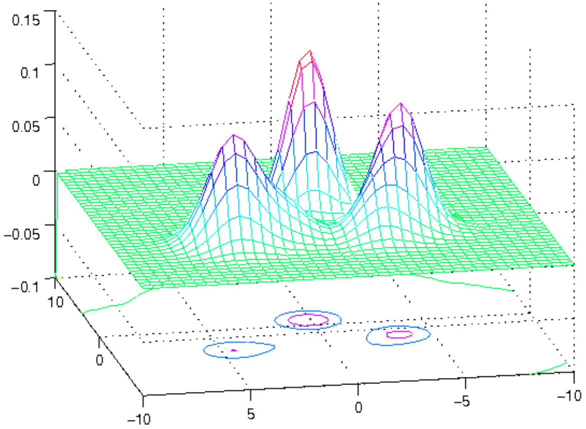


**Figure 1: A 2-dimensional Gaussian mixture model with three components. Contour lines below the GMM mark the one- and two- standard deviation ellipses.**

## 3. GAUSSIAN MIXTURE MODELS

The Gaussian probability density function (pdf) is the statistical distribution:

$$\mathcal{N}(x;\mu,\sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \qquad (1)$$

completely characterized by the mean $\mu$ and variance $\sigma^2$.

A Gaussian mixture model is a multimodal distribution resulting from a combination of several Gaussian components [24]. Figure 1 shows an example of a 2-dimensional GMM with three Gaussian components. The GMM can be characterized by the vector $\Theta$ of the means, variances, and weights of its $C$ components:

$$\Theta = \{\mu_1, \sigma_1, \omega_1, ...\mu_C, \sigma_C, \omega_C\} \qquad (2)$$

where the weight of each component is the portion of samples belonging to that component, such that $0 < \omega_c \leq 1$ and $\sum_{c=1}^{C} \omega_c = 1$.

The probability density function of a GMM, parametrized by $\Theta$, is defined as a weighted sum over its Gaussian components:

$$p(x|\Theta) = \sum_{c=1}^{C} \omega_c \mathcal{N}(x;\mu_c,\sigma_c) \qquad (3)$$

Several approaches exist for estimating the parameters of the GMM given a set of datapoints. The most popular, and the one used here, is the expectation-maximization (EM) algorithm [8], which iteratively optimizes the model using maximum likelihood estimates.

We elected to base our approach on Gaussian mixture models because of previously reported successes using classification methods for demonstration learning [4, 18]. GMMs provide a built-in measure of classification confidence, which is required for our confident execution approach, and are also robust to noise, generalize, and capture correlations between continuous features. All of these characteristics make GMMs a powerful tool for robotic data analysis.

# 4. CONFIDENCE-BASED LEARNING FROM DEMONSTRATION

In this section we present the details of our demonstration-based learning approach. Section 4.1 describes our algorithm for learning and using an action policy given a set of training data. Section 4.2 presents our confident execution approach for acquiring training data and focusing learning on relevant parts of the domain by adjusting the autonomy of the robot. Section 4.3 discusses two ways of applying confident execution to learning a new task.

Our approach utilizes the *learning by experienced demonstration* [15] technique, in which the robot is fully under the expert's control while continuing to experience the task through its own sensors. During each training timestep, the robot records sensory observations about its environment and executes the action selected by the human expert.

Observations are represented using an $n$-dimensional feature vector that can be composed of continuous or discrete values. The agent's actions are bound to a finite set $\mathcal{A}$ of action primitives, which are the basic actions that can be combined together to perform the overall task. Each labeled training point consists of the pair $(o, a)$, with observation $o$ and expert-selected action $a \in \mathcal{A}$. The goal is to learn the policy $\pi : o \longrightarrow \mathcal{A}$, mapping observations to action primitives.

## 4.1 Multi-Mixture Policy Learning

We use a set of Gaussian mixture models to generate and represent the action policy based on the training data. To learn the policy, all datapoints are separated into classes based on their action label, so that all observations leading to the same action are clustered together. Since a single action is often associated with a number of distinct domain states (the action *turn left* may be taken from several different locations), a separate Gaussian mixture is used to represent each action class. Components within the mixture represent distinct state regions, and each mixture may have a different number of components. Our policy is therefore represented by the set $\{\mathcal{G}_a : a \in |\mathcal{A}|\}$, where $\mathcal{G}_a$ is a GMM representing action $a$, parameterized by $\Theta_a$ with $C_a$ components, such that:

$$p_a(o|\Theta_a) = \sum_{c=1}^{C_a} \omega_c \mathcal{N}(o; \mu_c, \sigma_c) \qquad (4)$$

The parameters of each mixture model are learned using the EM algorithm. Since EM requires the number of components in the mixture as input, the algorithm uses the Akaike Information Criterion (AIC)[2] to find the optimal number of components per mixture. AIC represents a measure of the goodness of fit of an estimated statistical model, calculated by:

$$AIC_k = 2C_k - 2\mathcal{L}(\mathcal{G}_k) \qquad (5)$$

where $\mathcal{L}$ is the likelihood of $\mathcal{G}_k$. The preferred model, represented by the lowest AIC value, is the one that best explains the data with a minimum number of components. For a set of mixture models, the optimal parameters are determined by taking a weighted sum of the AIC values over the entire set:

$$AIC = \sum_{k=1}^{|\mathcal{A}|} AIC_k \Pr(\mathcal{G}_k) \qquad (6)$$

where $\Pr(\mathcal{G}_k)$ is the probability that a point is generated by mixture $k$ based on the portion of datapoints belonging to that mixture. Using the AIC, the optimal number of components is determined by permuting the number of components per mixture from one to some fixed maximum (bound by the number of datapoints in the mixture), and searching over this space to find the combination with the lowest AIC value. This search ranges from the assumption that all mixtures are represented by a single Gaussian, to the assumption that every point in each mixture forms its own component. Most models, however, have far fewer Gaussian components than datapoints, and so basic knowledge of the domain can be used to limit the maximum number of components to a small number to avoid extensive computation.

**Algorithm 1** Confident execution

```
$observation \leftarrow$ GetSensorData()
$(gmmAction, conf) \leftarrow$ Classify($observation$)
if $conf >$ autonThresh then
    ExecuteAction($gmmAction$)
else
    $expertAction \leftarrow$ GetExpertAction()
    LogDatapoint($observation, expertAction$)
    if $numNewDatapoints >$ maxNew then
        UpdateModel()
    ExecuteAction($expertAction$)
```

Once the algorithm learns the parameters of the model, it classifies a new datapoint by assigning it to the mixture class $m$ with the maximum likelihood:

$$m = \operatorname*{argmax}_{1 \le k \le |\mathcal{A}|} p_k(o|\Theta_k) \Pr(\mathcal{G}_k) \qquad (7)$$

The output of the classification is the action represented by the selected GMM. Additionally, the model returns a confidence value representing the certainty of the classification based on the likelihood.

The complete model can be used to control the agent's actions through direct execution of the learned policy. The agent follows the policy by classifying the current observation, and then deterministically executing the action selected by the model.

## 4.2 Confident Execution

In this section, we describe the method for acquiring the training data by adjusting the autonomy of the learning agent. Algorithm 1 presents a pseudocode summary of our algorithm.

At each learning timestep, the robot uses its learned model to classify its current observation of the environment. For each classification, the model returns a recommended action and a classification confidence. The agent selects between autonomously executing the action and requesting an expert demonstration by comparing the classification confidence value to an autonomy threshold parameter. Classification confidence greater than the threshold results in autonomous execution of the model-selected action by the agent, while confidence below the threshold interrupts the execution of the task while the agent waits for the expert to select the appropriate action. The agent's model is updated by relearning the mixture parameters after maxNew new datapoints are obtained.

As the agent performs its task, it alternates between autonomous execution in familiar domain states, and supervised demonstration in areas where further training data is needed. As it gathers more information about the domain, more of the agent's observations are classified with high confidence and fewer demonstrations become necessary.

The process of alternating between autonomous and supervised execution is referred to as adjustable autonomy, and this method has been proven effective in a wide range of robotic applications, from personal assistants [19] to space exploration [21]. Our algorithm combines learning with adjustable autonomy, resulting in an interactive teaching method that reduces dependence on the human expert over time.

Adjustable autonomy has a number of advantages over traditional, one-shot learning approaches. It reduces the workload of the human expert by eliminating repetitive demonstrations of already learned elements of the task. Additionally, by requesting demonstrations in low confidence situations the algorithm provides feedback to the expert, calling attention to areas of the domain where further learning is needed. As a result, in many domains the expert is able to adapt the training routine to focus on difficult to learn areas, reducing the overall learning time of the algorithm. Finally, partial autonomy can be used to restrict robot activity in complex dynamic environments where human or robot safety is a concern.

The confidence threshold value that determines the level of autonomy is set and adapted by the human operator or by the agent itself. Since the threshold value is continuous, our approach allows smooth adjustment of the autonomy level. We can also configure the algorithm to automatically adjust the confidence threshold based on the prediction accuracy of the model using a similar approach to the one presented in [17].

## 4.3 Confident Execution Application

Confident execution can be applied to learn a new task in one of two ways. One option is to begin learning directly with the confident execution approach, without an existing prior model. In this case, all initial classifications return a confidence of zero, and the agent always requests demonstration until the first model estimate is learned.

Alternatively, confident execution can be preceded by a non-interactive demonstration phase to build up an initial model. The expert may prefer this approach for complex tasks, where generating a long but continuous demonstration is simpler than a large number of interrupted ones (we found this to be the case for driving, see Section 5). Using this approach, the transition from non-interactive demonstration to confident execution is determined by the quality of the learned policy.

We evaluate the performance of the learned policy by how closely it matches the behavior of the expert. Prior to updating the model with a new training point $(o, a)$, we classify the observation $o$ using the current model. We then compare the model-selected action to the demonstrated action $a$. Performing this comparison over a window of consecutive training points results in an estimate of the prediction accuracy of the model that relates how closely the policy matches the behavior of the expert. This estimate enables the expert to track the agent's learning progress in real time, and to determine when model proficiency is high enough to switch to confident execution.

The entire learning process is completed when the agent is able to execute the task under full autonomy, or when the expert is satisfied with the performance of the model. At this point, the robot executes the learned policy directly, as described in Section 4.1.

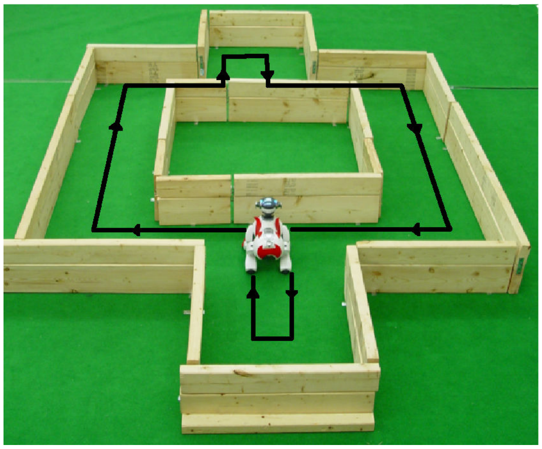


**Figure 2: The corridor navigation domain. The robot's target path is marked by a black line.**

# 5. EXPERIMENTAL RESULTS

We validate our approach by applying the learning framework to two domains. A real-world corridor domain is used with a Sony AIBO robot to demonstrate characteristics of the algorithm in a real, sensory-based environment. We then present results of a complex simulated driving task.

## 5.1 Corridor Navigation Domain

In the corridor navigation task, shown in Figure 2, the AIBO must navigate the domain in a circular path. The black line in the figure marks the navigation path demonstrated by the expert.

The robot observes the environment using the IR sensor built into the head. By turning its head, the robot calculates distance to the nearest obstacle in three directions – left, right and front – resulting in a 3-dimensional continuous feature vector. Due to the noise of the IR sensor, the robot processes all sensor readings when it is stationary and averages the values over 15 consecutive readings.

The robot has four available actions: *forward*, *turn left*, *turn right* and *u-turn*. The *forward* action moves the robot approximately 20 cm in the direction it is facing. *turn left* and *turn right* rotate the robot while slowly advancing it forward, and the *u-turn* action rotates the robot 180°. After completing each action, the robot stops to take the next sensor reading; it requires a minimum of 26 actions to complete one circuit of the domain. The goal of the learning is to classify each observation into one of the four action classes. To perform the task correctly, the robot must learn to distinguish between open space, nearby and far away walls.

Figure 2 shows the starting configuration of the robot. During the initial training phase, the robot was teleoperated by the expert using a wireless joystick until the learned model achieved 90% prediction accuracy over a 20-move window. Learning then transitioned to the confident execution phase. The robot required 46 consecutive demonstration

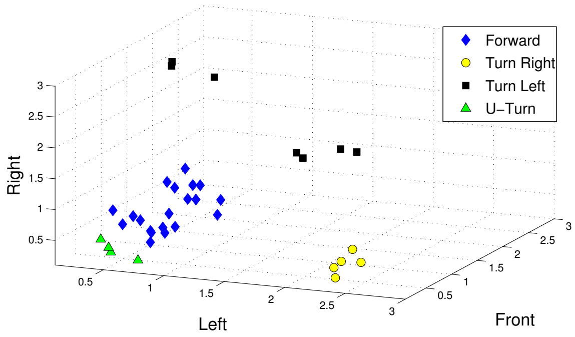


**Figure 3: Corridor navigation domain training data representing all of the action classes.**

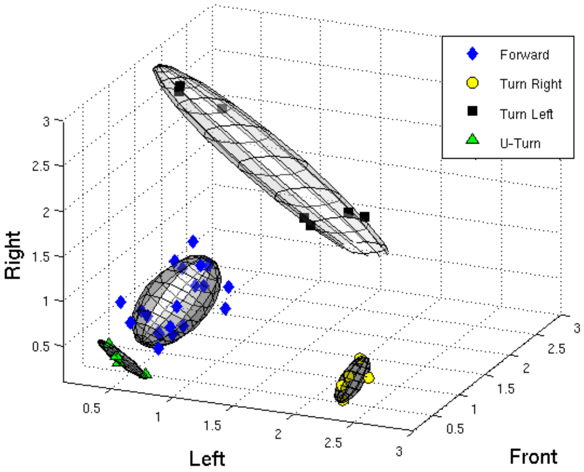


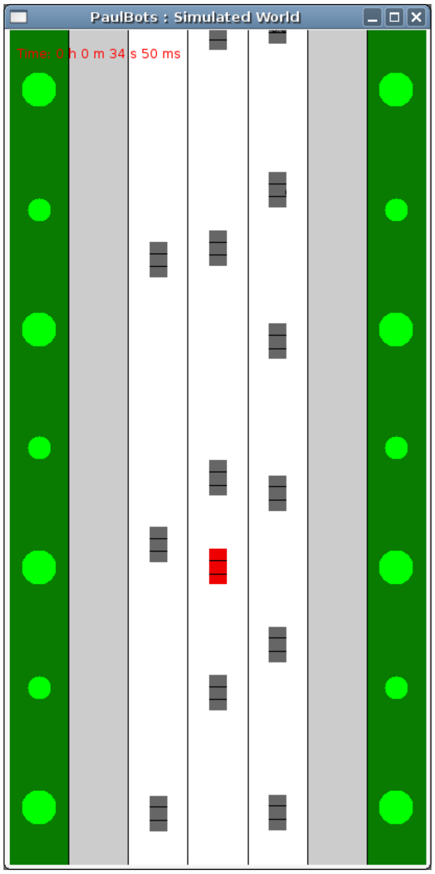
**Figure 4: Gaussian mixtures fitted to corridor navigation domain training data.**

steps, or just under two complete traversals of the environment, to complete this first learning phase.

Learning continued with the confident execution approach using a classification confidence threshold fixed at two standard deviations for the most likely Gaussian component. During the next circuit, the robot requested help at 3 out of 26 locations, followed by two more queries during the next two passes. The learning process was completed after the agent continued to navigate correctly and autonomously for an additional ten traversals of the domain.

Figure 3 shows data from the four Gaussian mixture models representing the final learned policy. In this domain, the action classes form clear independent clusters, resulting in non-overlapping Gaussian components as shown in Figure 4. Note that the *turn left* datapoints form two clusters as two different domain states map to this action. However, in this case the model uses a single Gaussian component to represent the data as this over-generalization does not affect classification of the other classes.

We compare the performance of our learning approach to reinforcement learning within the same domain. During the RL experiment, the robot learned the task by exploring the



Time: 0 h 0 m 34 s 50 ms

**Figure 5: Screenshot of the driving simulator.**

domain and learning the policy using the Prioritized Sweeping algorithm [13]. To avoid complex parameters that can affect RL performance in continuous domains, we simplify the domain by discretizing the real-values sensor data into binary wall/no-wall features typical of most grid-world experiments. Our results show that even with this advantageous simplification, RL takes significantly longer to learn the policy than our demonstration-based method while achieving the same performance. Specifically, using a reward function that returns greater reward values for less recently visited states (thereby encouraging circular movement through the domain), the Prioritized Sweeping algorithm took an average of 275 steps to learn the optimal policy, compared to the 53 demonstration steps required by our algorithm.

## 5.2 Driving Domain

We now present results of a challenging simulated car driving domain (Figure 5), inspired by a similar task introduced in [1]. In this domain, the agent takes the shape of a car that must be driven by the expert on a busy road. The agent travels at a fixed speed of 60 mph, while all other cars move in their lanes at predetermined speeds between 20-40 mph. Since the agent can not change its speed, it must navigate between other cars to avoid collision. The agent is limited to three actions: remaining in the current lane, and shifting one lane to the left or right of the current position. The road has three normal lanes and a shoulder lane on both sides;

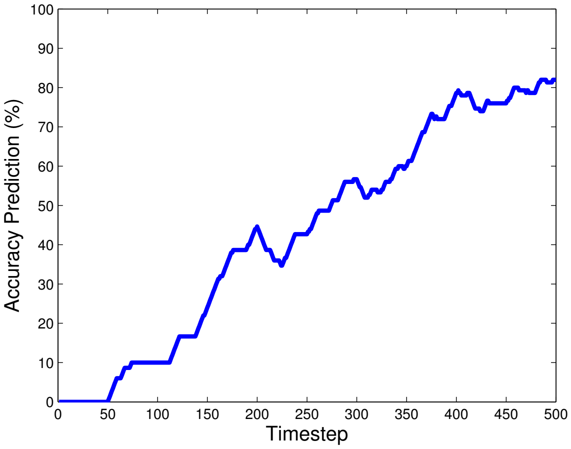


**Figure 6: Prediction accuracy of the learned model over the initial training phase. Prediction accuracy window = 150 timesteps.**

the car is allowed to drive on the shoulder but can not go off-road.

The environment is represented using four features, a distance to the nearest car in each of the three lanes and the current lane of the agent. The agent's lane is represented using a discrete value symbolizing the lane number. The distance features are continuously valued in the [-25,25] range; note that the nearest car in a lane can be behind the agent.

A human demonstrated the task by using a keyboard interface. Figure 6 shows the prediction accuracy of the model during the initial training phase. The expert continued to perform non-interactive demonstrations until the model reached 80% prediction accuracy over a 150-timestep window, which resulted in a demonstration length of 500 timesteps, or approximately 2.1 minutes. After transitioning to the confident execution phase, the expert concluded the training after 150 demonstration timesteps when the model exhibited good performance. During the confident execution phase, the expert performed all demonstrations as sequences of ten consecutive moves.

The final learned model consisted of 34 Gaussian components over three mixture models (one for each action class). The feature space of this domain is much more complex than that of the corridor domain as the different action classes frequently overlap. Figure 7 shows a sample of the data representing the policy for driving in the middle lane (Lane2). The data is split into two regions based on the relative position (in front or behind) of the nearest car in the agent's current lane. No samples appear in the 10 to -10 distance range along the Lane2 axis as the expert avoids collisions that would occur from having another car in such close proximity.

The final policy was able to imitate the expert's driving style and navigate well in the complex driving domain.[1] Additionally, our agent's performance is comparable to that achieved by Abbeel et al. in a similar domain using Inverse Reinforcement Learning [1].

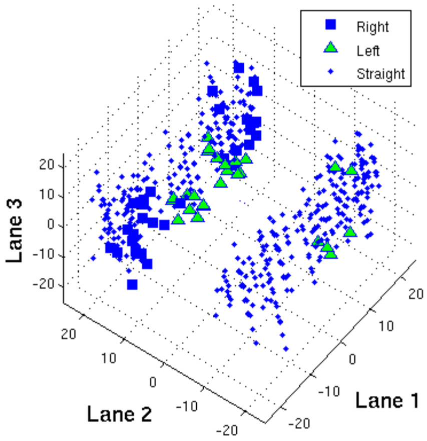


**Figure 7: Driving training data demonstrating the agent's driving policy in the middle lane. Graph axes represent distance to the nearest car in each of the three driving lanes.**

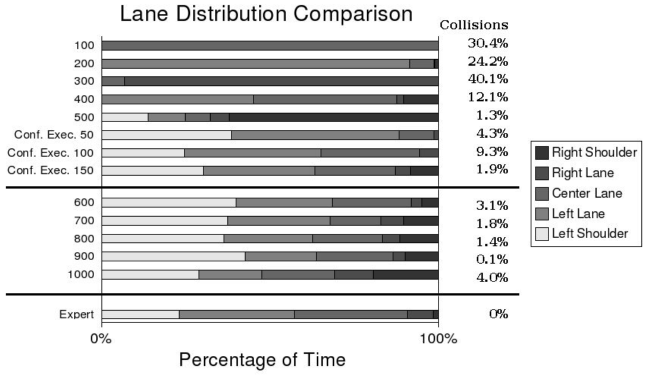


**Figure 8: Policy performance comparison using lane distribution and collision evaluation metrics.**

### 5.2.1 *Evaluation*

Since the algorithm aims to imitate the behavior of the expert, no 'true' reward function exists to evaluate the performance of a given policy. However, we present two domain-specific evaluation metrics that capture the key characteristics of the driving task.

Since the demonstrated behavior attempts to navigate the domain without collisions, our first evaluation metric is the number of collisions caused by the agent. Collisions are measured as the percentage of the total timesteps that the agent spends in contact with another car. Always driving straight and colliding with every car in the middle lane results in a 30% collision rate.

Our second evaluation metric is the proportion of the time the agent spends in each lane over the course of a trial. This metric captures the driving preferences of the expert and provides an estimate of the similarity in driving styles. Each evaluation trial was performed for 1000 timesteps over an identical road segment.

Figure 8 shows the agent's performance at different stages

[1] A video of the final learned policy is available at http://www.cs.cmu.edu/∼soniac

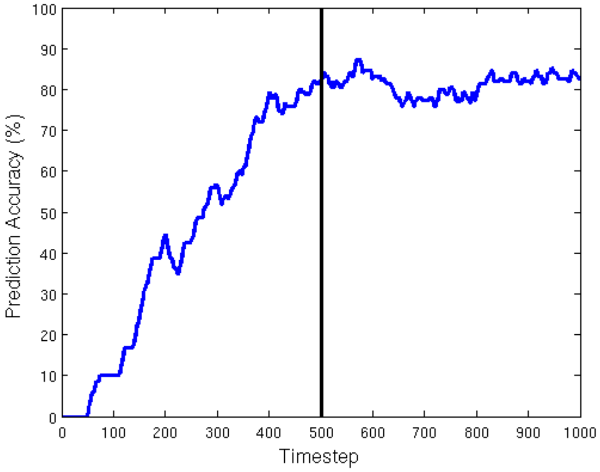


**Figure 9: Prediction accuracy over the entire non-interactive training sequence. The first 500 training trials are duplicated on the left side of the image.**

of the learning process with respect to these metrics. Each bar in the figure represents a composite graph showing the percentage of time spent by the agent in each lane. Collision percentages for each policy are reported to the right of the bar graphs. The bottom bar in the figure shows the performance of the expert over the evaluation road segment (not used for training). We see that the expert successfully avoids collisions, and prefers to use the left three lanes, only rarely using the right lane and right shoulder.

The top section of the graph, containing eight bars, summarizes the agent's performance throughout the non-interactive demonstration and confident execution learning phases. Initially the agent always remains in the center lane, accumulating a 30.4% collision rate in the process. As learning progresses, the agent learns to change lanes effectively, beginning to use all five available lanes after 500 demonstration instances, with a collision rate of only 1.3%. However, the agent's lane preference differs significantly from the expert as the agent spends most of its time avoiding traffic by driving on the right shoulder. The next three bars display performance during the confident execution phase at 50-timestep intervals. Similarity in lane preference improves over this final training phase, reaching final performance very similar to that of the expert with a low 1.9% collision rate.

### 5.2.2 *Comparison to Non-Interactive Demonstration Learning*

In this section, we compare the performance of the confident execution approach to training the model using only non-interactive demonstration. We replace the confident execution phase with 500 additional non-interactive demonstrations by the expert, resulting in a total of 1000 consecutive demonstrated timesteps. Figure 9 shows the prediction accuracy over the entire training time.

Note that despite the large number of demonstrations, the prediction accuracy does not continue to grow. This arises due to the complexity of the domain and the inconsistency of the human demonstration as frequently more than one action is reasonably applicable for a given situation. Consider for example the case in which the agent is approaching a car in its own lane. The agent must switch lanes in order to pass, but in addition to deciding the direction to turn, the agent must also decide at which timestep to do it. In reality, switching lanes can be successfully accomplished over a range of states, and a human demonstrator is unlikely to select the exact same instance every time. As a result, the model is not able to accurately predict the expert's actions and the prediction accuracy is only able to estimate the quality of the model to some degree.

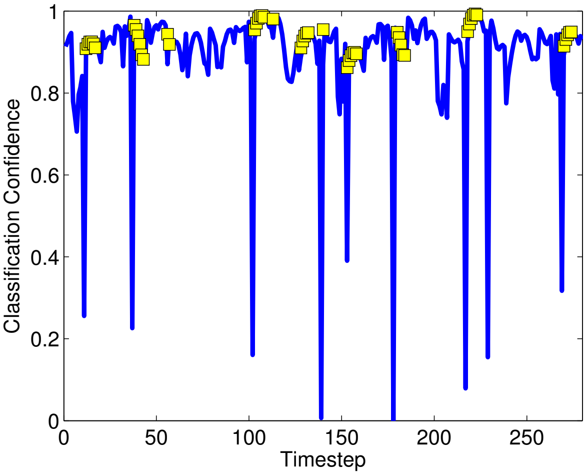


**Figure 10: Gaussian mixture distance values of the best classification over a sequence of timesteps. Squares mark timesteps during which the agent was involved in a collision.**

Five bars in the middle segment of Figure 8 show the learner's performance for the 500 additional non-interactive demonstration timesteps. The agent initially shows a strong preference for driving in the left shoulder, but achieves a final driving pattern that is again similar to that of the expert, although with a slightly higher collision rate of 4.0%. In summary, we find that the confident execution approach achieved slightly better performance while requiring several hundred fewer demonstration trials than non-interactive demonstration training.

Finally, we present evidence of why confident execution works to improve the learning process. Figure 10 plots a sequence of classification confidences reported by the model that was learned after the first 500 demonstrations. Squares plotted along the curve mark timesteps during which collisions with other cars occur. The majority of collisions occur shortly after a significant drop in classification confidence, indicating that the agent was in a situation of high uncertainty. Confident execution works not only by preventing the low confidence action from being executed and leading to a collision, but also by labeling the uncertain observation using the action selected by the expert during the resulting demonstration, increasing the likelihood that the correct action will be selected with high confidence in the future.

## 6. CONCLUSION

When creating an intelligent agent, it is often difficult to encode expert knowledge procedurally. Teaching by demonstration is a direct and intuitive approach aimed at addressing this problem. In this work, we presented an interactive policy learning approach, based on Gaussian mixture models, that reduces the number of required demonstrations by allowing an agent to actively request and effectively represent the most relevant training data. Using our approach, the agent operates with increasing autonomy as it achieves proficiency at its task, eliminating the need for repeated demonstrations of acquired behavior. Our results demonstrate that confident execution significantly reduces the number of training timesteps compared to exploration and non-interactive demonstration based methods, while achieving equivalent or superior performance levels.

## 7. ACKNOWLEDGMENTS

This research was partly sponsored by United States Department of the Interior under Grant No. NBCH-1040007 and by BBNT Solutions under subcontract no. 950008572. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of any sponsoring institution. We would also like to thank Paul Rybski for providing his simulator package.

## 8. REFERENCES

[1] P. Abbeel and A. Y. Ng. Apprenticeship learning via inverse reinforcement learning. In *International Conference on Machine learning*, New York, NY, USA, 2004. ACM Press.

[2] H. Akaike. A new look at the statistical model identification. *IEEE Transaction on Automatic Control*, 19:716–722, 1974.

[3] C. G. Atkeson and S. Schaal. Robot learning from demonstration. In *International Conference on Machine Learning*, pages 12–20, San Francisco, CA, USA, 1997. Morgan Kaufmann Publishers Inc.

[4] 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.

[5] D. C. Bentivegna, G. Cheng, and C. G. Atkeson. Learning from observation and from practice using behavioral primitives. *11th International Symposium of Robotics Research*, 2003.

[6] 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.

[7] C. Breazeal, G. Hoffman, and A. Lockerd. Teaching and working with robots as a collaboration. In *AAMAS '04: Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 1030–1037, Washington, DC, USA, 2004. IEEE Computer Society.

[8] A. Dempster, N.M.Laird, and D. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of Royal Statistical Society*, 8(1), 1977.

[9] G. Hayes and J. Demiris. A robot controller using learning by imitation. In *2nd International Symposium on Intelligent Robotic Systems*, 1994.

[10] T. Inamura, M. Inaba, and H. Inoue. Acquisition of probabilistic behavior decision model based on the interactive teaching method. In *Ninth International Conference on Advanced Robotics (ICAR)*, pages 523–528, 1999.

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

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

[13] A. W. Moore and C. G. Atkeson. Prioritized sweeping: Reinforcement learning with less data and less time. *Machine Learning*, 13(1):103–130, 1993.

[14] 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.

[15] M. N. Nicolescu and M. J. Mataric. Natural methods for robot task learning: instructive demonstrations, generalization and practice. In *Second International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 241–248, New York, NY, USA, 2003. ACM Press.

[16] J. Pearl. *Probabilistic Reasoning in Intelligent Systems*. Morgan Kaufmann.

[17] R. Ros and J. L. Arcos. Acquiring a robust case base for the robot soccer domain. In *International Joint Conferences on Artificial Intelligence*, 2007.

[18] J. Saunders, C. L. Nehaniv, and K. Dautenhahn. Teaching robots by moulding behavior and scaffolding the environment. In *HRI '06: Proceeding of the 1st ACM SIGCHI/SIGART conference on Human-robot interaction*, pages 118–125, New York, NY, USA, 2006. ACM Press.

[19] P. Scerri, D. Pynadath, and M. Tambe. Towards adjustable autonomy for the real world, 2003.

[20] S. Schaal, A. Ijspeert, and A. Billard. *Computational Approaches to Motor Learning by Imitation*, pages 199–218. Number 1431. Oxford University Press, 2004.

[21] M. Sierhuis, J. Bradshaw, A. Acquisti, R. Hoof, R. Jeffers, and A. Uszok. Human-agent teamwork and adjustable autonomy in practice, 2003.

[22] W. D. Smart and L. P. Kaelbling. Effective reinforcement learning for mobile robots. In *IEEE International Conference on Robotics and Automation*, 2002.

[23] A. L. Thomaz, G. Hoffman, and C. Breazeal. Reinforcement learning with human teachers: Understanding how people want to teach robots. In *15th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*, 2006.

[24] D. M. Titterington, A. Smith, and U. E. Makov. *Statistical Analysis of Finite Mixture Distributions*. Wiley, New York, NY, 1985.

[25] M. van Lent and J. E. Laird. Learning procedural knowledge through observation. In *K-CAP '01: Proceedings of the 1st international conference on Knowledge capture*, pages 179–186, New York, NY, USA, 2001. ACM Press.