ARL-TR-9626 ● JAN 2023

DEVCOM ARMY RESEARCH LABORATORY

Modeling Collision Avoidance Decisions by a Simulated Human-Autonomy Team

by Evan C Carter and Vernon J Lawhern

NOTICES

Disclaimers

The findings in this report are not to be construed as an official Department of the Army position unless so designated by other authorized documents.

Citation of manufacturer's or trade names does not constitute an official endorsement or approval of the use thereof.

Destroy this report when it is no longer needed. Do not return it to the originator.

ARL-TR-9626 ● JAN 2023

Modeling Collision Avoidance Decisions by a Simulated Human-Autonomy Team

Evan C Carter and Vernon J Lawhern
DEVCOM Army Research Laboratory

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED (From - To)
January 2023	Technical Report	10 January 2021–29 September 2022

4. TITLE AND SUBTITLE
Modeling Collision Avoidance Decisions by a Simulated Human-Autonomy Team

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)
Evan C Carter and Vernon J Lawhern

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
DEVCOM Army Research Laboratory
ATTN: FCDD-RLA-FA
Aberdeen Proving Ground, MD 21005

8. PERFORMING ORGANIZATION REPORT NUMBER
ARL-TR-9626

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release: distribution unlimited.

13. SUPPLEMENTARY NOTES
ORCID IDs: Evan C Carter, 0000-0001-7471-8769; Vernon J Lawhern, 0000-0002-3921-8723

14. ABSTRACT
Creating autonomous teammates that respond to their human counterparts in an adaptive way requires models of human goal and intent. One approach to building such models is inverse reinforcement learning, a process by which a reward function, representing goals and intent in a task, is learned from observed behavior. Here, we validate an approach to inverse reinforcement learning in a collision avoidance task completed by a simulated human-autonomy team. Our findings indicate several promising aspects of our approach, as well as some clear drawbacks for future work to investigate.

15. SUBJECT TERMS
machine learning, decision-making, human-autonomy teaming, obstacle avoidance, inverse reinforcement learning, Humans in Complex Systems

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Evan C Carter
Unclassified	Unclassified	Unclassified	UU	24	19b. TELEPHONE NUMBER (Include area code) (240) 478-9295

Standard Form 298 (Rev. 8/98)
Prescribed by ANSI Std. Z39.18

Contents

List of Figures

List of Tables

1. Introduction

A requisite step in creating artificially intelligent teammates that can adapt themselves to their human counterparts is developing computational modeling methods that can express the goals and intents of humans to AI systems. It is possible to achieve this goal with a wide variety of methods, from models that offer pure prediction from past data to generative models based only on theory. One promising approach is found in methods for so-called learning from demonstration (Argall et al. 2009; Ravichandar et al. 2020), a research thrust that takes demonstration data, such as behavior by an expert performing a task, and trains models (commonly referred to as "agents") to perform the task as the expert would. In this report, we adopt methods from the realm of learning from demonstration to model and predict a simulated robot's behavior in a collision avoidance teaming task. Specifically, we apply Inverse Reinforcement Learning (IRL) (Ng and Russell 2000; Arora and Doshi 2021), a method that infers a reward function from demonstrations.

The task, which is based on a research video game being used to study human-autonomy teaming (Adamson et al. 2017), involves a robot under the shared control of a human player and an AI agent. Without input from the player, the AI agent controls the robot, but the player can override the agent at any point, similar to a real-world case of driving with an automated driving assistant. This task presents a challenge to methods for learning from demonstration aimed at modeling human intent because observed behavior on the task results from a combination of control from two demonstrators: one human and one autonomous. For example, the human's behavior may be produced by both an understanding of their own goals and an estimate of the goals of the AI. Moreover, when the AI is in control, all that is known about the human is that they are refraining from providing input—the degree to which the human agrees with the choices by the AI is hidden.

Our focus on this particular task is motivated by ongoing work by our team to collect data from participants using the research video game that inspired the task. Ultimately, we will attempt to model behavior by real people over long time periods—playing daily for 180 days—in a way that facilitates the development of an adaptive AI agent. The work described here is a validation of a method that will move our team toward this goal; however, the approach is general enough in that its core concepts can be applied elsewhere.

2. Methods, Assumptions, and Procedures

2.1 The Task

The task space is a 2-D Cartesian plane over which the robot and 120 obstacles navigate. Movement by both robot and obstacles is determined by heading—angles from 1° to 360°—and speed—0.13 units every time step. Obstacles always move as a set at the start of a time step. The robot always moves after the obstacles.

At the beginning of a simulation run, all obstacles have their starting positions and headings randomly drawn from a uniform distribution, ranging from –20 to 20 for x- and y-coordinates and 1° to 360° for heading. Obstacles keep the same heading throughout the run and wrap around the task space once the Euclidian distance from their position to the origin is greater than 50. The robot's starting position is always at a point given by (25, 0) and its heading can change at each time step according to a controller.

At the start of a run, each task space is populated by a goal located at the origin. After the robot "collects" the goal (by reducing the distance between itself and the goal to less than 0.60), a new goal appears. Goals appear at a set sequence of pairs of points: (0, 0), (25, 25), (–25, –25), (0, 0), (25, –25), (–25, 25). The simulation run ends after the last goal is collected or a maximum number of time-steps N have been logged. All controllers are designed to approach goals and avoid obstacles; however, it should be noted that "colliding" with an obstacle does not end a simulation run.

2.2 Controllers

Two controllers are implemented in the task, and both make decisions on the basis of a repulsive and an attractive field. The repulsive field, r, is a function of the Euclidian distance, u, between the robot and all obstacles: $1/\sqrt{2\pi\sigma^2} - \exp(u^2/2\sigma^2)$. The attractive field, g, is defined simply as the Euclidian distance between the robot and the goal. Both controllers are specified by two parameters, the repulsion weight, α, and the deviation, σ, and both make decisions on the sum $r\alpha + g(1-\alpha)$. An example of the feature space is given in Fig. 1.

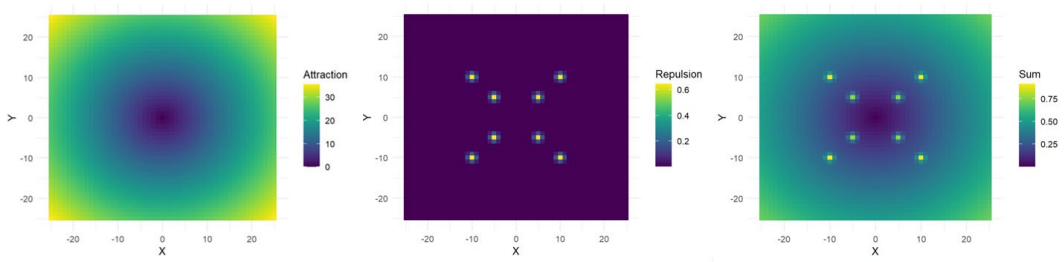

Fig. 1 Attractive and repulsive fields and their sum. A goal is placed at the origin and eight stationary obstacles are placed around the goal. For this example, $\sigma = 0.62$ and $\alpha = 0.98$.

As in the research game on which the task is based, the robot can be controlled following a Gaussian Artificial Potential Field (APF) (Weerakoon et al. 2014; Chiang et al. 2015). In this case, the controller follows the gradient of $r\alpha + g(1 - \alpha)$. Alternatively, the robot can be controlled by a Reinforcement Learning (RL) controller where σ and α represent a value function. The RL controller models the task as a first-exit, Linearly solvable Markov Decision Problem (LMDP) (Todorov 2006) with a single transition.

We define the state space of the LMDP as having a single internal state—the current position of the robot, and 360 boundary states—the 360 positions that result from moving one step in each direction given by headings of 1° to 360°. Each state is represented in the feature space given by r and g as defined.

In an LMDP, a decision-maker takes action by modifying the passive dynamics of the environment $p(x'|x)$—that is, the transition between the current state x and the next state x' that would occur if no decision-maker intervened. The cost function in this setting is

$$l\big(x, \pi(\cdot\,|x)\big) = q(x) + KL(\pi(\cdot\,|x)||p(\cdot\,|x))\,, \tag{1}$$

where $q(x)$ is the state-cost and Kullback–Leibler (KL) divergence represents the so-called cost of control—a cost that tracks the degree to which actions by the controller deviate from the passive dynamics. The modified Bellman equation for an LMDP is

$$z(x) = \exp\big(-q(x)\big)\, G[z(\cdot)](x)\,, \tag{2}$$

where $z(x)$ is the desirability function and G gives the expected z when following $p(x'|x)$ from x to all next possible x'. The optimal policy for this problem is given as

$$\pi^*(x'|x) = \frac{p(x'|x)z(x')}{G[z(\cdot)](x)}. \tag{3}$$

Solving the forward problem is an exercise in taking a known state-cost function and known passive dynamics and deriving an optimal policy given the previous. As mentioned, the RL controller considers only a single transition; therefore, the controller identifies the boundary state that maximizes $z(x)$ and then moves to that new position.

In this project, we make no attempt at solving this forward problem by training an RL controller to complete the task. Instead, we focus on the inverse problem (i.e., IRL) where a trajectory of N state transitions $\tau = \{x_t, x'_t\}_{t=1,\ldots,N}$ are provided by an APF controller and our goal is to learn the value function, $v(x)$, an RL controller would follow to produce the same behavior. Under the assumptions of an LMDP, the value function is a direct transformation of the desirability function, $z(x) = \exp(-v(x))$, which, as defined, is a function of the state-cost, $q(x)$, and the passive dynamics.

As in many IRL applications, Dvijotham and Todorov (2010) proposed that the value function be represented as a weighed sum of features:

$$v(x) = \sum wf(x). \tag{4}$$

As mentioned, we define the features as the repulsive and attractive fields, r and g, and the weights as α and $1 - \alpha$, respectively. Thus, the value of a given state depends on the distances between the robot and the other objects in the environment and the controller's preferences as represented by σ and α. Given that an estimated value function and estimated desirability function can be calculated, and the RL controller can perform the task by maximizing the estimated desirability function.

2.3 Likelihood-Free Inference

Dvijotham and Todorov (2010) define a negative log-likelihood from the optimal policy, $\pi^*(x'|x)$, which can be used for a likelihood-based IRL method. Our initial experimentation with this approach produced poor performance, due seemingly to arithmetic underflow issues that resulted from $p(x'|x)$ being very small (e.g., if $p(x'|x)$ was a random walk from the internal to the boundary states, it was always $\frac{1}{360} = 0.003$).

As a solution to this problem, we implemented a simple method of Approximate Bayesian Computation (ABC) (Csilléry et al. 2010; Turner and Van Zandt 2012, 2018). ABC is one of many versions of so-called likelihood-free inference in which parameter estimates can be produced when no likelihood is available, but a data simulator is. Unlike the method given by Dvijotham and Todorov (2010), parameter estimates given by ABC are represented as distributions over parameters

(approximations of a Bayesian posterior). This is an advantage because, in principle, many combinations of parameters can produce the same behavior, a phenomenon known as policy invariance (Balakrishnan et al. 2020) in the IRL literature. If a method provides only a point estimate for parameters in the value function, it can lead to a mistaken understanding that only a specific parameter combination is consistent with the data.

Furthermore, some trajectories can be considered as allowing better discrimination between estimated value functions. Imagine, for example, a trajectory in our task in which no obstacles are present. In this case, all controllers with $\alpha < 1$ would produce identical behavior—heading directly towards the goal. A method that provides results that reflect the fact that, given the data, one cannot distinguish between many possible estimates is more valuable than a method that hides uncertainty.

In ABC, a candidate parameter value, θ_i, is sampled from the prior distribution and fed to the simulator. The simulated output is then compared to the observed output and a distance value, d_i, is calculated. In this report, we make use of a class of ABC methods that apply a similarity kernel to link posterior probability and the distance value (Turner and Van Zandt 2018). Here, we assume a discretized, 2-D, uniform prior over σ and α with steps of 0.1 and limits at $[0.05, 1.4]$ and $[0.05, 1]$, respectively. Thus, the prior is a grid of 140 unique pairs $\theta_i = \{\sigma_i, \alpha_i\}_{i=1,\dots,140}$. Given an input of a state, x_t, at time step, t, all values of the prior are fed to the RL controller and an estimated next state, $\hat{x}'_{ti}$, is observed for each value. The Euclidian distance in task space between $\hat{x}'_{ti}$ and the observed x'_t is used as d_{ti}. Thus, at each time step, t, ABC returns a distribution, d_t, of 140 distance values.

We study two kernels that relate d_t to p_t. First, is a standard Gaussian

$$p_t = 1/\sqrt{2\pi\sigma_{d_t}^2} - \exp\left({d_t}^2/\sigma_{d_t}^2\right), \tag{5}$$

where $\sigma_{d_t}^2$ is the standard deviation of the distribution, d_t, and the values are then normalized to sum to one. The second kernel is an Exponential kernel

$$p_t = \exp\left(-\gamma d_t\right), \tag{6}$$

where $\gamma = 10$. Again, values are normalized to sum to one.

An example of a set of five posteriors calculated for five time steps with the Gaussian kernel is shown in Fig. 2.

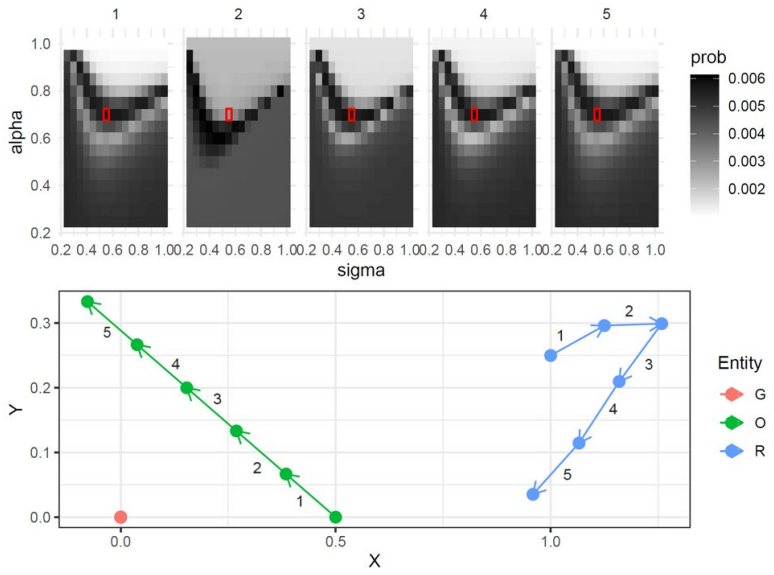

Fig. 2 IRL method. Top panel: Posteriors probability distributions calculated using our ABC method with a Gaussian kernel. Each distribution is associated with a time step one through five. The red squares in the space indicate the true parameters that generated the data. Bottom panel: The corresponding task space with goal (G), obstacle (O), and robot (R) positions over time.

2.4 Simulation Experiment Design

Our goal was to assess whether our IRL method could accurately estimate the ground-truth parameters of an APF controller given a set of observed transitions. To do this, we ran three experiments.

In each experiment, four APF controllers (see Table 1) were defined identically to the APF agents used in our previous work. These choices were made to produce a wide variety of behavior in terms of the trade-off of completing the task quickly versus safely. Ten sets of starting obstacle positions were randomly generated, and each AI was recorded in a simulation run initialized with each set of obstacles. Simulation runs terminated after either the AI collected all goals or $N = 2{,}000$ time steps were logged.

Table 1 Ground truth APF controllers

	AI 1	AI 2	AI 3	AI 4
α	0.72	0.98	0.75	0.75
σ	0.55	0.62	1.40	1.00

In Experiments 2 and 3, the APF controller was modified so that a secondary APF controller would take over in certain conditions. The purpose of this modification was to mimic human behavior observed in the research game—specifically, that in

some cases the human only takes control of the robot for very short periods of time (Adamson et al. 2017). Given that our goal is to learn the goals and preferences of the human, it is critical that our IRL method be able to extract multiple value functions when behavior is produced by a mixture of controllers. We set the secondary APF controller to have the same σ as the primary controller but to have $\alpha = 1$ and $\alpha = 0.01$ in Experiments 2 and 3, respectively. These values produce AIs that either only focus on avoiding obstacles ($\alpha = 1$) or only focus on moving toward the goal ($\alpha = 0.01$).

In both Experiments 2 and 3, the secondary controller took control from the primary controller when the distance between the robot and any obstacles was less than 1. Once the secondary controller was engaged, it stayed active for 10 time steps and then control was given back to the primary controller.

We assessed our results in terms of the coefficient of determination, R^2, between the x- and y-coordinates associated with the observed states and the states estimated by the IRL controller defined by the θ that maximized the posterior. This approach can be conceptualized as a means of evaluating the posterior predictive distribution—a distribution of estimated positions that would result from data generated by draws of θ from the posterior. We refer to this value as the maximum posterior predictive, R^2.

The maximum posterior predictive, R^2, was calculated using the multivariate version of R^2 defined by Jones (2019). Both the 1-D and multidimensional R^2 coefficients are given as $1 - SS_{res}/SS_{tot}$. In the typical 1-D case, $SS_{tot} = \sum_{t=1}^{N}(y_t - \bar{y})^2$, or the sum of squared differences between the observed values and the mean of the observed values, and $SS_{res} = \sum_{t=1}^{N}(\hat{y}_t - y_t)^2$, or the sum of squared differences between a model's estimate and the observed values. In the multidimensional case, rather than a simple difference, a Euclidian distance is taken, and rather than a single value of $\bar{y}$, the average point in space is used. For all versions of R^2, values of one indicate perfect fit of the data and values of zero indicate that the model is no better than a random guess. Unlike the 1-D R^2, however, the multidimensional version can also take negative values (i.e., SS_{res} is bigger than SS_{tot}), which indicate that the mean point in space is closer to the observed values than those estimated by the model.

In addition to the maximum posterior predictive, R^2, we assess the full posterior predictive distributions. Our approach produces a posterior for every time step in all 10 simulations; therefore, we have many posterior predictive distributions. To aid in interpretation, and to make our assessments comparable across time steps, we translate task space to a standardized space in which all observed transitions move from the origin to (0, 0.13) and all estimated transitions move from the origin

to a point in the positive y-range (i.e., quadrants I and II) with a Euclidian distance of 0.13 units from the origin. The posterior predictive distribution can then be given over the angle, ε, formed at the origin between the observed and estimated transitions. The best possible value of ε is zero, indicating that correspondence between the observed and estimated transitions implied by the posterior is exact. The worst possible value of ε is 180, indicating that the estimated transition implied by the posterior is exactly opposite the observed transition. These distributions provide a sense of the consistency between the observed data and those implied by the posteriors.

To assess the quality of parameter estimation our approach provides, we calculate the Shannon entropy of every posterior at every time step, $h_t = -\sum_i^{140} p_i log_2\,(p_i)$. Given that the posterior comprises a grid of 140 cells, the maximum value of $h_t = 7.129$, which corresponds to each candidate value of θ having equal posterior probability. Thus, lower values of h_t indicate greater certainty in which values of θ are implied by the observed behavior. Due to policy invariance, certain environmental configurations will correctly produce maximum entropy—recall the previous example of a trajectory where no obstacles are present.

3. Results and Discussion

3.1 Maximum Posterior Predictive R^2

Figures 3 and 4 show the R^2 results. These results are given separately for the primary and secondary controllers—that is, R^2 values are calculated for data that has been subset by which controller produced the behavior. As a result, no values exist for the secondary controller in Experiment 1 (because the secondary controller was not available) and results from Experiments 2 and 3 represent far fewer data points because the secondary controller only took control in specific situations. To further illustrate this last point, Fig. 5 shows R^2 values as a function of the percent of time in which the primary controller was engaged.

When examining the primary controller alone in Figs. 3 and 4, two patterns stand out the most. First, AI 2 is difficult to learn, showing mostly negative R^2 values. Second, the Exponential 10 kernel seems to provide the best overall performance. Note, for example, the difference in R^2 values between kernels for AI 1 in Experiment 2.

Additionally, Fig. 3 suggests that the secondary controller results in lower R^2 values, particularly in Experiment 2 where R^2 values become negative in many cases. This is particularly striking for the difference between the two kernels for

AI 1: R^2 values go from almost entirely negative with the Gaussian kernel to primarily positive for the Exponential 10 kernel.

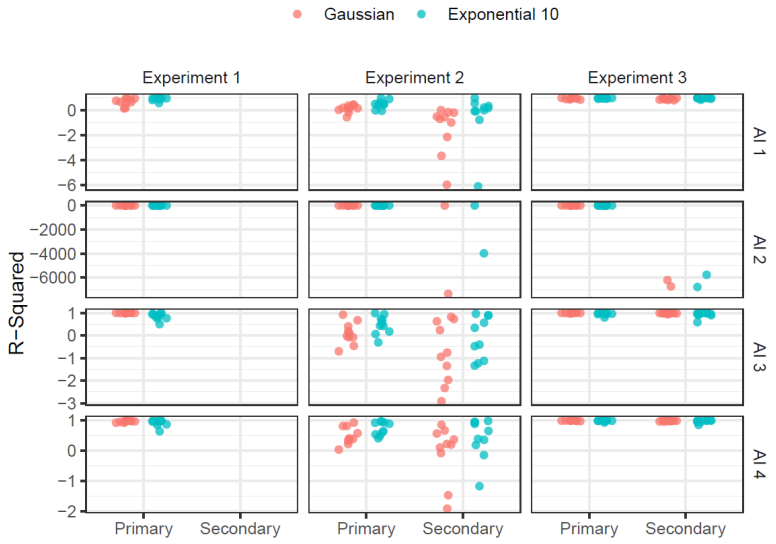

Fig. 3 Maximum posterior predictive R^2 shown for each simulation run, by each AI, in each experiment. The R^2 are further divided into groups by kernel and which controller produced the behavior (note that only the primary controller was available in Experiment 1).

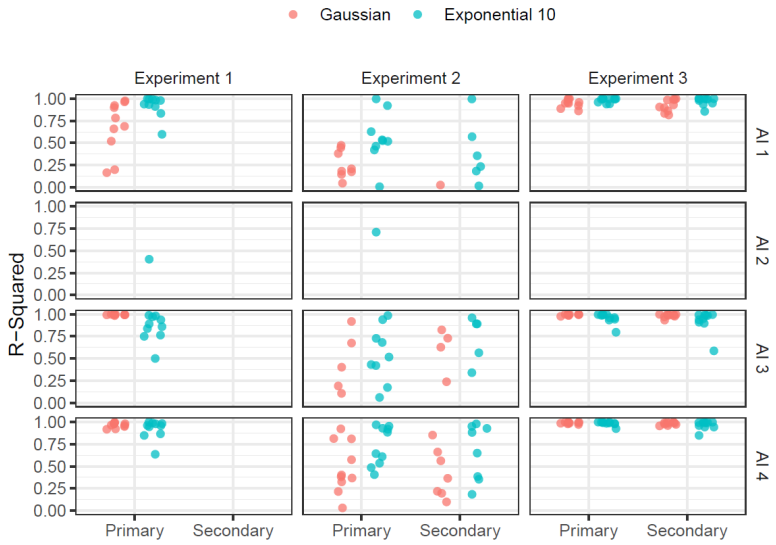

Fig. 4 Maximum posterior predictive R^2 shown for each simulation run, by each AI, in each experiment. The R^2 are further divided into groups by kernel and which controller produced the behavior. This figure is identical to Fig. 3, but only shows the positive values of R^2.

The results in Fig. 5 suggest that the primary controller produces the vast majority of data (i.e., typically greater than 90%). Additionally, it appears that different AI end up in situations in which the secondary controller takes over less frequently than others. Note, for example, AI 3's significantly greater dispersion along the

horizontal in Fig. 5 as compared to other AI. For some AI in Experiment 2, it also appears that a positive correlation exists between R^2 and the percent of time the primary controller is engaged. This relation suggests that the behavior of the secondary controller implemented in Experiment 2—one that only prioritizes obstacle avoidance—is much harder to learn than the secondary controller implemented in Experiment 3.

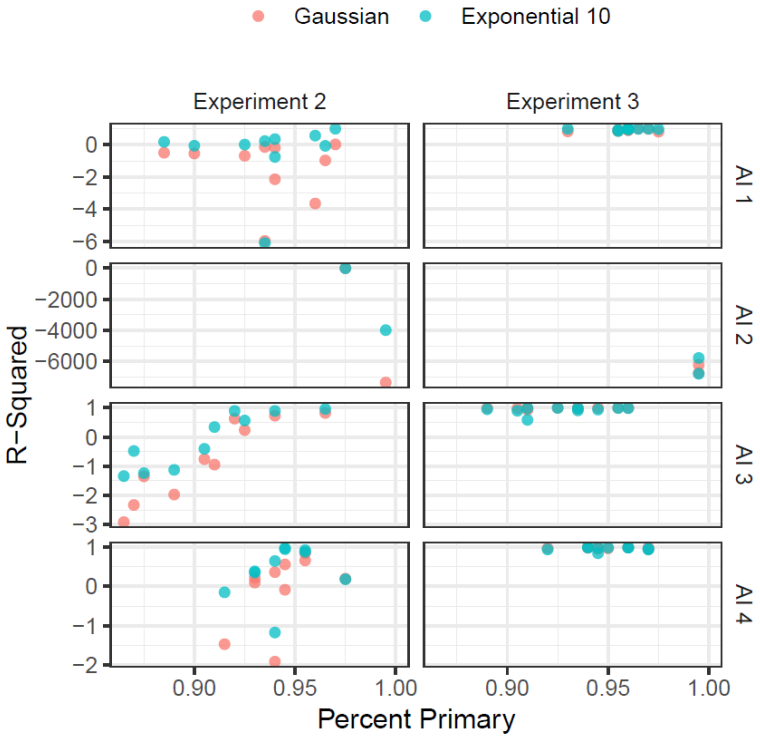

Fig. 5 Maximum posterior predictive R^2 by the percentage of time in which the primary controller produced behavior in each simulation run. This relation is show for each AI in each experiment and broken up by kernel.

3.2 Posterior Predictive Distributions Over ε

Figures 6 and 7 show the posterior predictive distributions over ε, the angle formed at the origin between the observed and estimated transitions when all transitions are normalized so that all transitions go from the origin to a point in the positive y-range. Recall that the R^2 in Figs. 3 and 4 is a function of the posterior predictive distributions' maximums; therefore, findings related to ε will be highly consistent. We present the full posterior predictive distribution as a means of an alternative assessment of how similar to the observed behavior the estimated behavior implied by the posteriors would be. It is therefore not surprising that a value of $\varepsilon = 0$ is given far higher probability in the posterior predictive distributions than other

values of ε in all cases, suggesting that our method tends to produce posteriors that are in tight correspondence to the observed data.

Some additional patterns are worth mentioning. First, the Gaussian kernel tends to assign higher probability to higher values of ε, and in some cases, even assigns large spikes of probability to values that suggest behavior that is nearly opposite of what was observed. Additionally, Experiment 2 and AI 2 both appear to result in distributions that are less consistent with the observed behavior, overall. Again, these findings are reflected in Figs. 3 and 4, as would be expected.

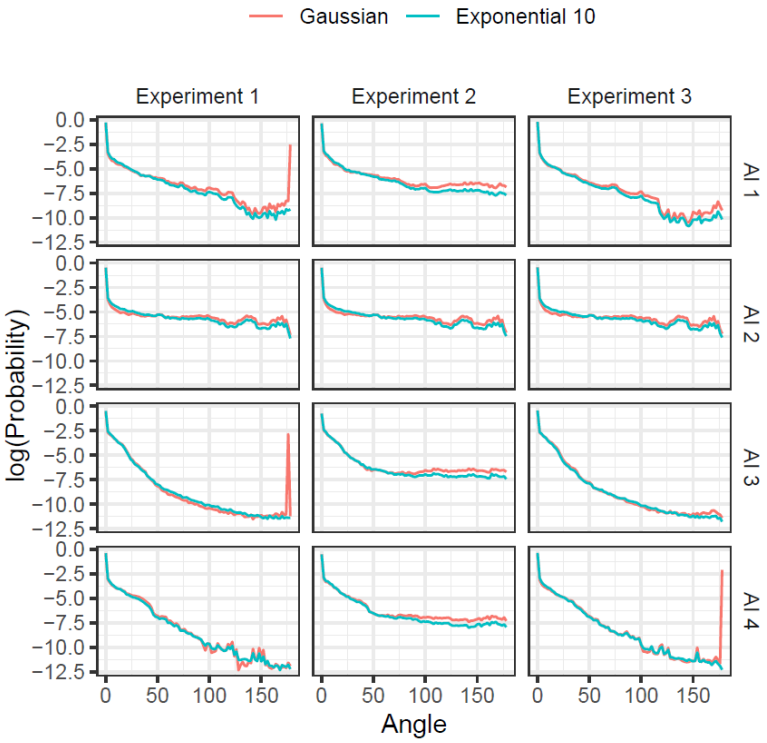

Fig. 6 The posterior predictive distributions over angle ε for all simulation runs given as log probability for ease of visualization. These distributions are separated out by AI in each experiment and given for both kernels.

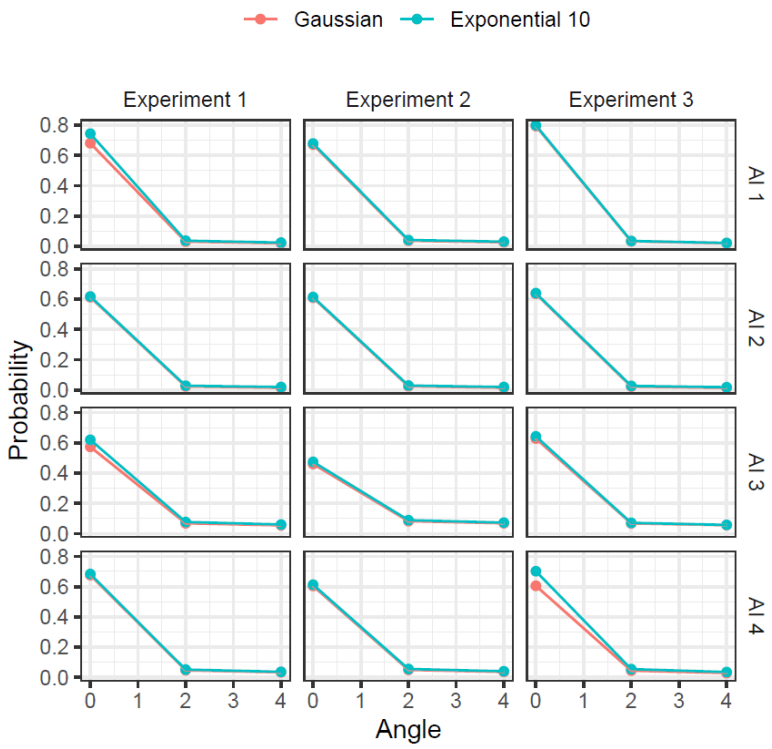

Fig. 7 The posterior predictive distributions over the first five angles ε for all simulation runs. Note that angles are binned by twos, starting with zero. These distributions are separated out by AI in each experiment and given for both kernels.

3.3 Entropy

Figures 8 and 9 display the entropy values for all posterior distributions. Figure 8 shows histograms of all values of entropy whereas Fig. 9 shows a time series of entropy values for each posterior assessed at each time point in the first simulation run.

As mentioned, entropy of these distributions can be 7.129 at most, which would indicate all candidate values of θ being equally likely (i.e., no new information is gained from observing the data). These values are highly related to the results for the posterior predictive distributions but offer a particular view of how specific the method's parameter estimation can be. Notably, the method does not appear to regularly provide precise answers.

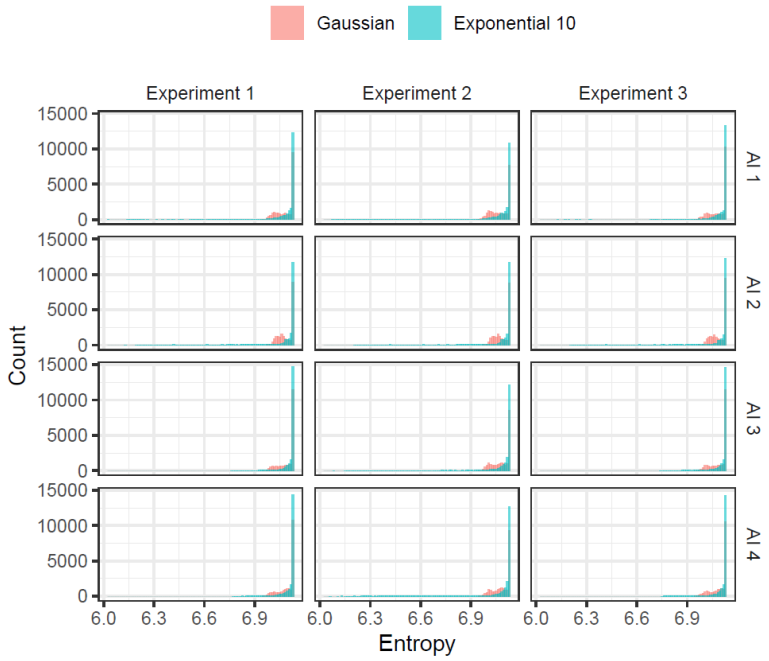

Fig. 8 Histograms of entropy values for every posterior across all simulation runs. Values are shown for each AI in each experiment and separated out by kernel.

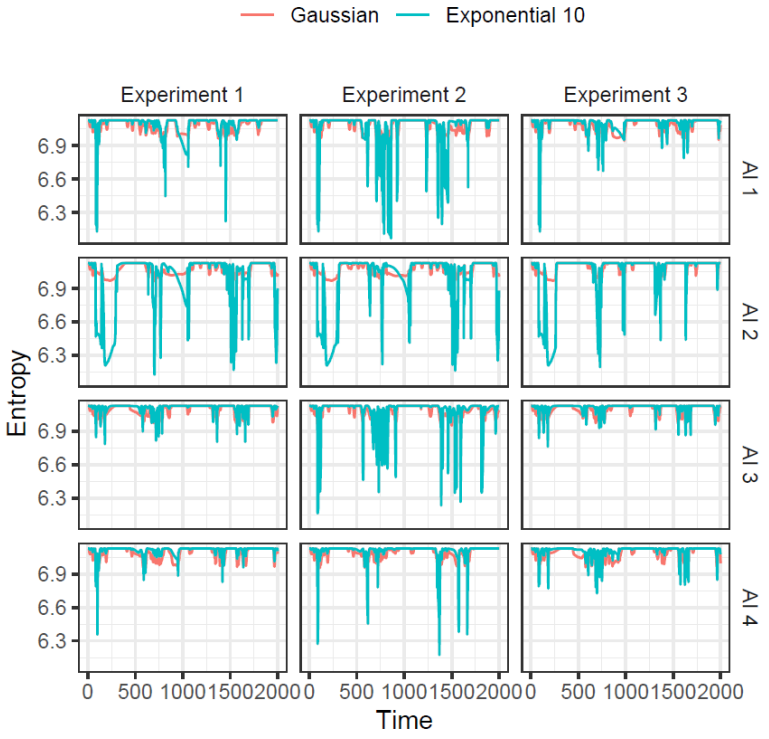

Fig. 9 Entropy of the posterior at every time step in the first simulation run. Time series are separated out by AI in each experiment and displayed for both kernels.

4. Conclusions

Our approach to modeling collision avoidance decisions in a collaborative task shows several promising results. Under some conditions, results look excellent (e.g., as seen in Fig. 4, Experiments 1 and 3 consistently have greater than 50% of variance explained); however, there are several areas where further investigation is needed. In particular, an analysis of when and why model fitting fails. AI 2 was difficult to model in all cases, and results from Experiment 2 suggest modeling the secondary controller might be especially difficult when the greatest preference is given to obstacle avoidance.

In addition to the areas that require further analysis, there are limitations to our findings that come from our assumptions and the design of our experiments. First among these is the fact that our LMDP is formulated in a way that very closely tracks the way in which the simulated demonstrators make decisions. When faced with human data, or even data from a different source, it is not clear that our method will perform well.

Secondly, our method of approximating posterior distributions happens on a per-time-step basis and no information between time steps is used. This was done to account for the potential change of controllers at every time step. Of course, the exact number of time steps over which a posterior should be calculated is dependent on the time scale at which the secondary controller (or the human) behaves. One future possibility is to adopt some method for change point detection as a means of deciding whether a given time step is included in the fitting process.

In our view, the main advantage of our approach is its simplicity. Although the implementation described here is highly specific, it should, in principle, be relatively easy to modify the method for new applications. Additionally, there are many more sophisticated ways to conduct ABC, and the adoption of recent advances might improve future uses of the approach we describe here.

5. References

Adamson T, Oishi M, Chiang HT, Tapia L. Busy Beeway: a game for testing human-automation collaboration for navigation. Proceedings of the 10th International Conference on Motion in Games; 2017. p. 1–6.

Argall BD, Chernova S, Veloso M, Browning B. A survey of robot learning from demonstration. Robotics and Autonomous Systems. 2009;57(5):469–483.

Arora S, Doshi P. A survey of inverse reinforcement learning: challenges, methods and progress. Artificial Intelligence. 2021;297:103500.

Balakrishnan S, Nguyen QP, Low BKH, Soh H. Efficient exploration of reward functions in inverse reinforcement learning via Bayesian optimization. Advances in Neural Information Processing Systems. 2020;33:4187–4198.

Chiang HT, Malone N, Lesser K, Oishi M, Tapia L. Path-guided artificial potential fields with stochastic reachable sets for motion planning in highly dynamic environments. Proceedings of the IEEE Intl Conf Rob Auto (ICRA); 2015. p. 2347–2354.

Csilléry K, Blum MG, Gaggiotti OE, François O. Approximate Bayesian computation (ABC) in practice. Trends in ecology & evolution. 2010;25(7):410–418.

Dvijotham K, Todorov E. Inverse optimal control with linearly-solvable MDPs. In: International Conference on Machine Learning; 2010.

Jones T. A coefficient of determination for probabilistic topic models. arXiv preprint arXiv:1911.11061. 2019.

Ng AY, Russell S. Algorithms for inverse reinforcement learning. In: ICML, Vol 1; 2000. p. 2.

Ravichandar H, Polydoros AS, Chernova S, Billard A. Recent advances in robot learning from demonstration. Annu Rev Control Robot Auton Syst. 2020;3:297–330.

Todorov E. Linearly-solvable Markov decision problems. Adv Neural Inf Process Syst. 2006;19.

Turner BM, Van Zandt T. A tutorial on approximate Bayesian computation. J Math Psychol. 2012;56(2):69–85.

Turner BM, Van Zandt T. Approximating Bayesian inference through model simulation. Trends Cogn Sci. 2018;22(9):826–840.

Weerakoon T, Ishii K, Nassiraei AAF. Dead-lock free mobile robot navigation using modified artificial potential field. Proceedings of the Joint 7th International Conference on Soft Computing and Intelligent Systems (SCIS); 2014 Dec. p. 259–264. doi:/10.1109/SCIS-ISIS.2014.7044812.

List of Symbols, Abbreviations, and Acronyms

1-/2-D	one-/two-dimensional
ABC	Approximate Bayesian Computation
AI	artificial intelligence
APF	Artificial Potential Field
IRL	Inverse Reinforcement Learning
KL	Kullback–Leibler
LMDP	Linearly solvable Markov Decision Process
RL	Reinforcement Learning

1 (PDF)	DEFENSE TECHNICAL INFORMATION CTR DTIC OCA
1 (PDF)	DEVCOM ARL FCDD RLB CI TECH LIB
2 (PDF)	DEVCOM ARL FCDD RLA FA E CARTER FCDD RLA FE V LAWHERN