A Bayesian Nonparametric Approach to Modeling Motion Patterns

Joshua Joseph · Finale Doshi-Velez · Albert S. Huang · Nicholas Roy

Abstract The most difficult—and often most essential—aspect of many interception and tracking tasks is constructing motion models of the targets to be found. Experts can often provide only partial information, and fitting parameters for complex motion patterns can require large amounts of training data. Specifying how to parameterize complex motion patterns is in itself a difficult task.

In contrast, nonparametric models are very flexible and generalize well with relatively little training data. We propose modeling target motion patterns as a mixture of Gaussian processes (GP) with a Dirichlet process (DP) prior over mixture weights. The GP provides a flexible representation for each individual motion pattern, while the DP assigns observed trajectories to particular motion patterns. Both automatically adjust the complexity of the motion model based on the available data. Our approach outperforms several parametric models on a helicopter-based car-tracking task on data collected from the greater Boston area.

Keywords Bayesian nonparametric modeling · markov decision process · interception and tracking

J. Joseph
E-mail: jmjoseph@mit.edu

F. Doshi-Velez
E-mail: finale@mit.edu

A. Huang
E-mail: ashuang@alum.mit.edu

N. Roy
E-mail: nickroy@mit.edu

Massachusetts Institute of Technology
77 Massachusetts Avenue, 33-315
Cambridge, MA 02139
Tel.: 617-253-2517
Fax: 617-253-7397

1 Introduction

The success of interception and tracking tasks often hinges on the quality of the motion models our agent has for predicting the target's future locations. These predictions are especially important when our agent's sensor range is limited. Unfortunately, the motion patterns of targets are often difficult to specify from expert knowledge alone. For example, suppose that our agent is a helicopter that must intercept and track a car or several cars in a large region such as a city. A model of traffic patterns may be hard to specify. Even determining what parameters are important to model the target's behavior—and how they should interact—can be a challenging task.

A data-driven approach to learning the target's motion patterns avoids the need for an expert to fully specify the target's motion model. Instead, the agent simply uses previously observed trajectories of the target to predict the target's future locations, where these predictions may depend on both the target's current position and past position history. Using a data-driven approach also side-steps the need to understand the target's motivations, which may appear irrational to an outside observer. For example, drivers rarely take the minimum-time route to a location; an expert model that assumes that optimizing travel time is the driver's primary objective will likely make poor predictions about a car's future locations. Our approach focuses on the features our own agent needs: good predictions of where the targets will be.

While a data-driven approach reduces the need for expert knowledge, the parameters of the motion model still need to be fit. We must also specify the class of models to which we expect the target's motion patterns to belong. For example, we may choose to model the target's motion as a series of straight-line segments, higher-order splines or even cylindrical trajectories. When considering real-world data, the correct class of motion models is not always obvious.

One solution is to consider sophisticated model classes with many parameters governing the forms of all the motion patterns we expect to occur. Unfortunately, while such a model class may be able to model the target's motion pattern, large amounts of data will be needed to train the many parameters. In many real-world domains, collecting sufficient data to train a large number of parameters may be prohibitively expensive.

In our previous work (Joseph et al, 2010), reviewed in section 3, we showed that nonparametric approaches to modeling motion patterns are well-suited for poorly-understood environments because they let the data determine the sophistication of the model—we no longer need to specify which parameters are important. Moreover, the Bayesian aspect helps the model generalize to unseen data and make inferences from noisy data.

Specifically, we can model a target's motion patterns with a Dirichlet process mixture model over Gaussian process target trajectories (DPGP). Using this nonparametric model boosts learning rates by generalizing quickly from small amounts of data but then adding sophistication as more target trajectories are observed. Previously (Joseph et al, 2010), we applied this DPGP model to applications tracking a single target whose current position was always observed (imagine having a GPS tracker on the target but not knowing where the target will go).

In this paper we present two key extensions to that previous work. First, we no longer assume that the target's position is available to the agent. Instead, we consider scenarios in which the agent can only observe the target if it is nearby; now the agent's goal is to first intercept and then track the target. Adapting our approach to make predictions about unseen targets using only partial information is one of our main contributions. Second, we also consider scenarios where multiple targets must be intercepted and tracked. Modeling multiple targets fits seamlessly into our DPGP model, demonstrating both the quality and versatility of our approach.

The remainder of this article is organized as follows: in section 2, we describe our DPGP motion model in detail. Section 3 reviews the utility of using the DPGP approach for tracking single agents whose current positions are always observed. We then demonstrate our extensions in applying our approach to multi-agent interception and tracking scenarios in section 4. Sections 5 and 6 discuss the scenarios in which we expect the DPGP model to perform well and place it in the context of prior tracking and interception literature.

2 Motion Model

We represent a target's trajectory t^i as a set of xy-locations $\{(x^i_1, y^i_1), (x^i_n, y^i_n), \ldots, (x^i_{L^i}, y^i_{L^i})\}$, where L^i is the length of trajectory t^i. Depending on how the trajectory data is collected, these locations may come at irregular intervals: for example, the distance between (x^i_t, y^i_t) and (x^i_{t+1}, y^i_{t+1}) may not be the same as the distance between (x^i_{t+1}, y^i_{t+1}) and (x^i_{t+2}, y^i_{t+2}). Trajectories may also be of different lengths L^i, both because some trajectories may be physically longer than others and because some trajectories may have a larger number of observed locations along the route.

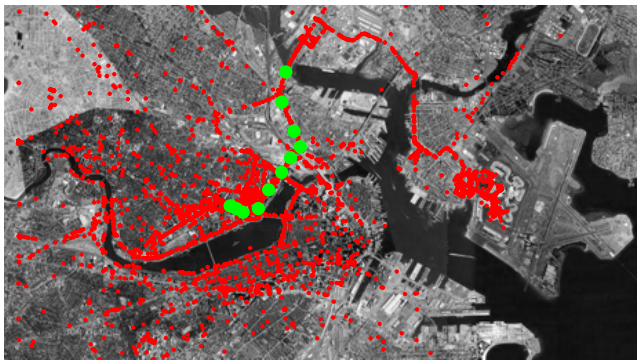


Fig. 1 A small set of the raw GPS data points (red) and a single trajectory (green) used to learn our model.

Throughout the paper we use time-stamped GPS coordinates of greater-Boston taxis from the CarTel project as our motivating dataset.[1] Figure 1 plots some of the trajectories (red points) on a map of Boston[2], emphasizing the discrete nature of our observations. One sample trajectory is highlighted in green, showing how the discrete observations are irregularly spaced along the trajectory. Working with trajectories of differing lengths is one of the challenges of this dataset, which we address by using Gaussian processes for modeling trajectories.

Motion Pattern We define a *motion pattern* as a mapping from locations to a distribution over trajectory derivatives $(\frac{\Delta x}{\Delta t}, \frac{\Delta y}{\Delta t})$ indicating the agent's future motion.[3] Given the target's current position (x_t, y_t) and a trajectory derivative $(\frac{\Delta x_t}{\Delta t}, \frac{\Delta y_t}{\Delta t})$, its predicted next position (x_{t+1}, y_{t+1}) is $(x_t + \frac{\Delta x_t}{\Delta t}\Delta t, y_t + \frac{\Delta y_t}{\Delta t}\Delta t)$. Thus, modeling trajectory derivatives is equivalent to modeling trajectories. In addition to being blind to the lengths and discretizations of the trajectories, modeling motion patterns as flow fields rather than single paths also allows us to group target trajectories sharing key characteristics: for example, a single motion pattern can capture all the paths that a target might take from different starting points to a single ending location.

[1] CarTel project, `http://cartel.csail.mit.edu`. The data was down-sampled to a rate of 1 reading per minute and pre-processed into trajectories based on if the car had stayed in the same place for a long enough time to indicate the end of a trajectory.

[2] `http://maps.google.com`

[3] The choice of Δt determines the scales we can expect to predict the target's next position well, making the trajectory derivative more useful than instantaneous velocity.

While velocity fields help group trajectories with certain characteristics, we still expect to encounter trajectories with qualitatively different behaviors. For example, different trajectories often share some segments but then branch off in different directions. These motion patterns are not well modeled by traditional techniques such as Markov chain models that simply try to predict a target's future location based on its current position (and ignoring its previous history). Returning to the CarTel taxi dataset, we see scenarios with overlapping paths are common: Figure 2 shows just one example of two routes that share a common corridor. In figure 2 the red trajectory travels east and the green trajectory travels north. We address this issue by using mixture models over motion patterns.

Mixtures of Motion Patterns A finite mixture model with M motion patterns $b_1, b_2, \ldots, b_M$ first assigns a prior probability for each pattern $p(b_1), p(b_2), \ldots, p(b_M)$. Given these prior probabilities, the probability of the i^{th} observed trajectory t^i under the mixture model[4] is

$$p(t^i) = \sum_j^M p(b_j) p(t^i|\theta_j) \tag{1}$$

where θ_j contains the parameters for motion pattern b_j.

The primary complication with a simple finite mixture model is that M is not known in advance, and may grow as more data is observed. Instead, we use a Dirichlet process mixture model, which allows us to create an infinite mixture of motion patterns. An important property of this model is that it places a prior over an infinite number of motion patterns such that the prior probabilities $p(b_1), p(b_2), p(b_3), \ldots$ still sum to one; the probability of a trajectory is

$$p(t^i) = \sum_j^\infty p(b_j) p(t^i|\theta_j). \tag{2}$$

This probability, and the number of different motion patterns in a given dataset, are determined during the inference process.

Motion Model We define the *motion model* as a mixture of weighted motion patterns. Each motion pattern is weighted by its probability and is modeled as a pair of Gaussian processes mapping (x, y) locations to distributions over trajectory derivatives $\frac{\Delta x}{\Delta t}$ and $\frac{\Delta y}{\Delta t}$. We place a Dirichlet process prior over mixture weights. Our DPGP motion model, detailed in the remainder of this section, is similar to models described by Rasmussen and Ghahramani (2002) and Meeds and Osindero (2006); however, unlike these previous works, our goal is to cluster trajectories of varying lengths, not just partition single points.

[4] Note that throughout the paper a t with a superscript, such as t^i, refers to a trajectory and a t without a superscript is a time value.

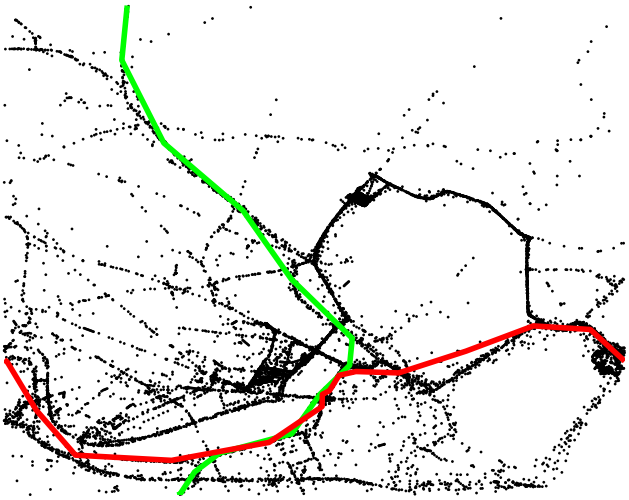


Fig. 2 An example of two trajectories that share a road segment. The red trajectory travels east and the green trajectory travels north. The Markov model cannot distinguish the two trajectories once they cross, but the DP model classifies them as two different paths.

Under our DPGP model, the prior probability of motion pattern b_j is given by its DP mixture weight π_j. The posterior probability of b_j given a target trajectory t^i is proportional to $\pi_j \cdot l(b_j; t^i)$, where $l(b_j; t^i)$ describes the likelihood of motion pattern b_j under trajectory t^i:

$$l(b_j; t^i) = \prod_t^{L^i} p\left(\frac{\Delta x_t}{\Delta t} \middle| x^i_{1:t}, y^i_{1:t}, \{t^k : z_k = j\}, \theta^{GP}_{x,j}\right) \cdot \prod_t^{L^i} p\left(\frac{\Delta y_t}{\Delta t} \middle| x^i_{1:t}, y^i_{1:t}, \{t^k : z_k = j\}, \theta^{GP}_{y,j}\right) \tag{3}$$

where z_k indicates the motion pattern to which trajectory t^k is assigned, and $\theta^{GP}_{x,j}$ and $\theta^{GP}_{y,j}$ are the hyperparameters of the Gaussian process for motion pattern b_j. Equation 3 may be applied to trajectories with differing numbers of observations or even trajectories that are only partially complete, which is particularly important when we wish to determine a target's motion pattern given only a few observations.

2.1 Gaussian Process Motion Patterns

Observations from a target's trajectory represent a continuous path through space. The Gaussian process (GP) places a distribution over functions (Rasmussen and Williams, 2005), serving as a non-parametric form of interpolation. Gaussian process models are extremely robust to unaligned, noisy measurements and are well-suited for modeling the continuous paths underlying our non-uniformly sampled time-series samples of the target's locations.

The Gaussian process for a motion pattern that models a trajectory's derivative is specified by a set of mean and

covariance functions. We describe the mean functions as $E[\frac{\Delta x}{\Delta t}] = \mu_x(x, y)$ and $E[\frac{\Delta y}{\Delta t}] = \mu_y(x, y)$, and implicitly set both of them to initially be zero everywhere (for all x and y) by our choice of parameterization of the covariance function. This encodes the prior bias that, without any additional knowledge, we expect the target to stay in the same place. Zero-mean GP priors also simplify computations.

We denote the covariance function in the x-direction as $K_x(x, y, x', y')$, which describes the correlations between trajectory derivatives at two points (x, y) and (x', y'). Given locations $(x_1, .., x_k, y_1, .., y_k)$, the corresponding trajectory derivatives $(\frac{\Delta x_1}{\Delta t}, .., \frac{\Delta x_k}{\Delta t})$ are jointly distributed according to a Gaussian with mean $\{\mu_x(x_1, y_1), .., \mu_x(x_k, y_k)\}$ and covariance Σ, where the $\Sigma_{ij} = K(x_i, y_i, x_j, y_j)$. In this work, we use the standard squared exponential covariance function

$$K_x(x, y, x', y') = \sigma_x^2 \exp\left(-\frac{(x - x')^2}{2{w_x}^2} - \frac{(y - y')^2}{2{w_y}^2}\right) + \sigma_n^2 \delta(x, y, x', y') \quad (4)$$

where $\delta(x, y, x', y') = 1$ if $x = x'$ and $y = y'$ and zero otherwise. The exponential term above encodes that similar trajectories should make similar predictions. The length-scale parameters w_x and w_y normalize for the scale of the data. The σ_n-term represents within-point variation (e.g., due to noisy measurements); the ratio of σ_n and σ_x weights the relative effects of noise and influences from nearby points. We use $\theta^{GP}_{x,j}$ to refer to the set of hyperparameters σ_x, σ_n, w_x, and w_y associated with motion pattern b_j (each motion pattern has a separate set of hyperparameters).[5]

For a GP over trajectory derivatives trained with tuples $(x_k, y_k, \frac{\Delta x_k}{\Delta t})$, the predictive distribution over the trajectory derivative $\frac{\Delta x}{\Delta t}^*$ for a new point (x^*, y^*) is given by

$$\mu_{\frac{\Delta x}{\Delta t}^*} = K(x^*,y^*,X,Y)K(X,Y,X,Y)^{-1}\frac{\Delta X}{\Delta t} \quad (5)$$
$$\sigma^2_{\frac{\Delta x}{\Delta t}^*} = K(x^*,y^*,X,Y)K(X,Y,X,Y)^{-1}K(X,Y,x^*,y^*)$$

where the expression $K_x(X, Y, X, Y)$ is shorthand for the covariance matrix Σ with terms $\Sigma_{ij} = K_x(x_i, y_i, x_j, y_j)$. The equations for $\frac{\Delta y}{\Delta t}^*$ are equivalent to those above, using the covariance K_y.

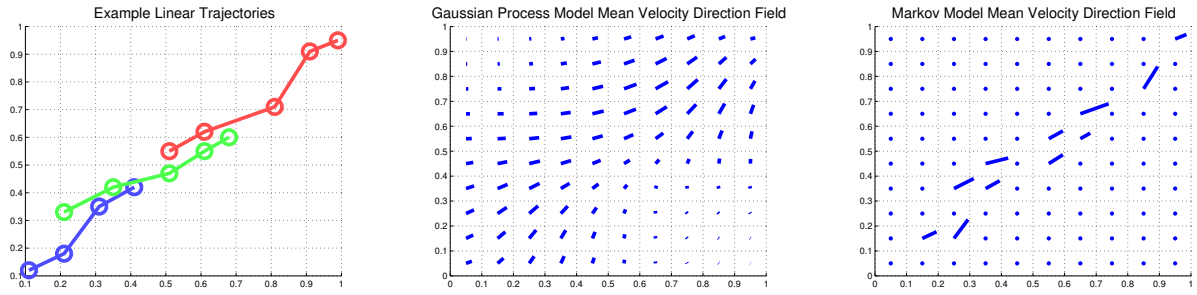
Estimating Future Trajectories As summarized in equation 5, our Gaussian process motion model places a Gaussian distribution over trajectory derivatives $(\frac{\Delta x}{\Delta t}, \frac{\Delta y}{\Delta t})$ for every location (x, y). In our prior work (Joseph et al, 2010), we used a simple approach to sample a target's possible trajectory multiple steps into the future: starting with the target's current location (x_1, y_1), we sampled a trajectory derivative $(\frac{\Delta x_1}{\Delta t_1}, \frac{\Delta y_1}{\Delta t_1})$ to get a next location (x_2, y_2). Then starting from (x_2, y_2), we sampled a trajectory derivative $(\frac{\Delta x_2}{\Delta t_2}, \frac{\Delta y_2}{\Delta t_2})$ to get a next location (x_3, y_3). We repeated this process until we had sampled a trajectory of length L. The entire sampling procedure was repeated from the current location (x_1, y_1) multiple times to get a sampling of future trajectories the target may take.

While samples drawn from this procedure are an accurate representation of the posterior over trajectories, sampling N trajectories of where the target may be L steps in the future requires NL queries to the Gaussian process. It also does not take advantage of the unimodal, Gaussian distributions being used to model the trajectory derivatives. Key to efficiently predicting future trajectories in this work is applying an approximation of Girard et al (2003) and Deisenroth et al (2009) that provides a fast, analytic approach of approximating the output of a Gaussian process given a distribution over the input distribution. In our case, our Gaussian process motion model over trajectory derivatives gives us a Gaussian distribution over possible target next-locations at each time step. The approximation of Girard et al (2003) and Deisenroth et al (2009) allows us to string these distributions together: we input a distribution of where the target may be at time t and a distribution of trajectory derivatives to get a distribution of where the target may be at time $t+1$. By being able to estimate the target's future trajectories analytically, we reduce the computations required—only L queries to the Gaussian process are needed to predict the target's location L steps into the future—and avoid the variance introduced by sampling future trajectories.

Comparison with a Markov chain model Instead of using a Gaussian process—which defines a distribution over velocities in a continuous state space—we could imagine a model that discretizes the state and velocity space into bins and learns a transition model between state-velocity bins. We call this alternative the "Markov model" because predictions about the target's next position depend only on the target's current position and velocity, not its past history.

A key question when trying to train such a Markov model is the appropriate level of discretization for the state space. In figure 3, we consider modeling a motion pattern that consists of approximately linear trajectories observed at irregular intervals. By modeling the velocity field over the continuous space, the GP is able to quickly generalize the velocity field over region, whereas the Markov model has gaps induced by its discretization. These gaps could be filled by a coarser discretization; however, the modeling would also be coarser. The GP automatically adjusts the generalization as more data arrive.

[5] We described the kernel for two dimensions, but it can be easily generalized to more.



(a) Example Trajectories (b) GP Mean Velocity Field (c) MM Mean Velocity Field

Fig. 3 Velocity fields learned by a GP and a Markov Model from three trajectories of an approximately linear motion pattern. The GP generalizes quickly from the irregularly observed trajectories, whereas the discretization in the Markov model slows down generalization.

2.2 Dirichlet Process Mixture Weights

Although a single Gaussian process can robustly model the variation within many closely related trajectories, it is not able to capture differences resulting from targets with different destinations, preferred routes, etc. To model qualitatively different motion patterns, we can represent the distribution over behaviors as a mixture of Gaussian processes. But, we cannot know ahead of time how many behaviors are sufficient for the model. To account for this ambiguity, we use Dirichlet processes.

The Dirichlet process is a distribution over discrete distributions in which the number of motion patterns is potentially unbounded, but with the expectation that there are a few patterns the target tends to follow most of the time.[6] If z_i indicates the motion pattern to which trajectory t^i is assigned, the prior probability that target trajectory t^i belongs to an existing motion pattern b_j is

$$p(z_i = j | z_{-i}, \alpha) = \frac{n_j}{N - 1 + \alpha}, \tag{6}$$

where z_{-i} refers to the motion pattern assignments for the remaining trajectories, α is the concentration parameter of the Dirichlet process, n_j is the number of trajectories assigned to motion pattern b_j, and N is the total number of observed trajectories. The probability that trajectory t^i exhibits a new motion pattern is

$$p(z_i = M + 1 | z_{-i}, \alpha) = \frac{\alpha}{N - 1 + \alpha}. \tag{7}$$

where M is the number of observed motion patterns.

Equation 7 implies that the number of motion patterns can grow as more data is obtained. This property is key to realistically modeling targets: the more interception and tracking tasks we perform, the more varieties of target motion patterns we expect to encounter. Figure 4 shows how the number of motion patterns grows (under our model) as new trajectories are observed for the actual dataset of greater Boston taxi routes (see section 3 for more details on the dataset). We show in section 3 that we can efficiently plan even when the number of actively observed motion patterns is unknown; moreover, this flexibility yields significantly improved results in the performance of the planner.

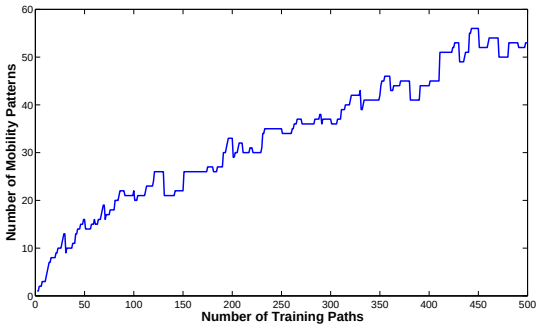


Fig. 4 As expected, the number of motion patterns in the taxi dataset increases as more trajectories are added.

DP Trajectory Classifying Example Just as the Gaussian process in section 2.1 allows us to model motion patterns without specifying a discretization, the Dirichlet process mixture model allows us to model mixtures of motion patterns without specifying the number of motion patterns. One could, of course, simply search over the number of motion patterns: we could train models with different numbers of patterns, examine how well each mixture model explains the data, and finally choose the best one. However, it is easy to show that this search requires much more computation time than using a Dirichlet process to automatically determine the number of patterns, with similar performance.

We compare the DPGP to a set of finite mixture models that also use Gaussian processes to model motion patterns (that is, the finite mixture model first described in equation 2). We consider the helicopter-based tracking scenario for a data set of taxi trajectories. Each model was trained on a batch of 200 trajectories using five different initializations. We tested tracking performance on a set of 15 held-out test

[6] See Teh (2007) for an overview of Dirichlet processes.

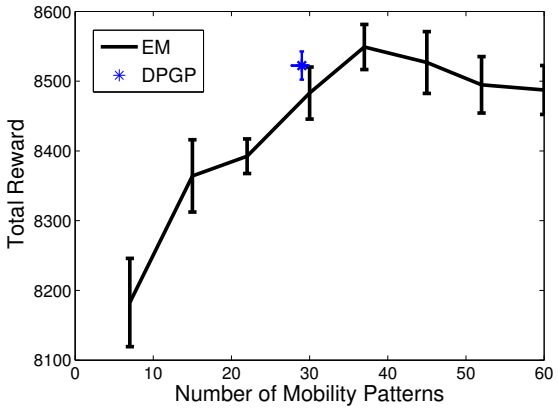


Fig. 5 Performance on 15 held-out test trajectories vs. model size for a variety of finite models (black) and the DPGP (blue) trained on 200 trajectories. The error bars represent the standard deviation of the reward from five runs. Note the inferred DPGP model has model size error bars also due to variation in the estimated model size for each run.

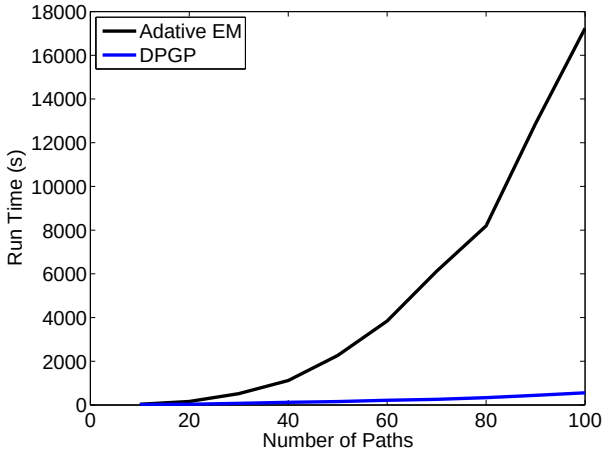


Fig. 6 Run time vs. number of paths for adaptive EM and our DPGP model.

trajectories. None of the models were updated during the testing phase.

The results in figure 5 show that while the finite GP-based models perform well overall, our DPGP model has nearly the best performance *without having to perform a search over the finite model space*. This last point is important, not only because a search over finite models would require more computation but also because the search requires us to choose a regularization criterion to avoid overfitting. Standard criteria, such as the Bayesian information criterion (Raftery, 1986) cannot be applied in this context because the GP contains an unbounded number of parameters; thus we must choose from various cross-validation or bootstrap procedures. The DPGP provides a principled, simple-to-use regularization criterion within its model.

Searching in the space of finite models is especially computationally expensive when the data arrives online and the number of clusters are expected to grow with time. (The DP can update the number of clusters incrementally.) To gain insight into the extra computation cost of this search process we implemented EM where every 10 paths we search over models sizes that are within five clusters of the current model. Figure 6 shows run time as the number of training paths increase for our DPGP model and this adaptive EM technique. The running time grows exponentially longer for EM with model search compared to the DPGP.

3 Application of Tracking with Full Information

We first consider the case in which our agent has access to the target's current position but needs to be able to predict its future position to track it effectively. We call this the "full information" case because this scenario implies that the agent has access to sensors covering the environment such that the target's current state is always known (up to time discretization). For example, we may be given location information from a dense sensor network. In this section, we formalize the tracking problem and describe the process of training a motion model for this full-information tracking task. We next provide results for our tracking problem applied to two targets with completely different motion models, one synthetic and one built from a real-world dataset. In Section 4, we will relax the assumption of a dense sensor network, and show how to extend our approach to target interception given information from a sparse sensor network.

3.1 Tracking Problem Formulation

Since the target's current position is known at every time step, we can formalize the scenario as a Markov decision process (MDP), a common tool for autonomous decision making. An MDP is defined by a set of states, a set of actions, a transition function, and a reward function. Here, the state is the joint position of our agent and the target $(x^a, y^a, x^{target}, y^{target})$. Given an action and our agent's current position (x^a_t, y^a_t), we assume that our agent's next position (x^a_{t+1}, y^a_{t+1}) is deterministic and known. In contrast, the target's transitions are stochastic over the continuous space; we can only place a distribution over the target's next position (x^a_{t+1}, y^a_{t+1}) based on our motion model. At each step, our agent incurs some small cost for moving, and receives a large positive reward each time it shares a grid cell with the target. Given an MDP, we can find the optimal policy using standard forward search techniques (Puterman, 1994).

3.2 Model Inference

Given a set of target trajectories, we can train the DPGP model from section 2 and use it to make predictions about future trajectories. Since exact inference over the space of DPs and GPs is intractable, we describe a process for drawing samples from the posterior over motion models. These samples are then used by our agent for planning.[7]

3.2.1 Training the Model

Our model contains two sets of parameters—the DP mixture weights π_k, the motion pattern assignments z_i, and the DP hyperparameter α—the GP hyperparameters $\theta^{GP}_{x,j}, \theta^{GP}_{y,j}$ and the trajectories assigned to each motion pattern cluster. Following the work of Rasmussen and Ghahramani (2002) and Rasmussen (2000), learning the model involves Gibbs sampling the parameters (see Algorithm 1).

We first resample each z_i in turn, using the exchangeability properties of the DP and GP to model the target trajectory t^i as the most recently observed target. The probability that the trajectory t^i will be assigned to an instantiated motion pattern is

$$p(z_i = j | t^i, \alpha, \theta^{GP}_{x,j}, \theta^{GP}_{y,j}) \propto l(b_j; t^i) \left(\frac{n_j}{N - 1 + \alpha} \right) \quad (8)$$

where $l(b_j; t^i)$ is the likelihood of motion pattern b_j from equation 3 and n_j is the number of trajectories currently assigned to motion pattern b_j. The probability that the trajectory t^i belongs to a new motion pattern is given by

$$p(z_i = M+1 | t^i, \alpha) \propto \int l(b_j; t^i) d\theta^{GP}_{x,j} d\theta^{GP}_{y,j} \left(\frac{\alpha}{N - 1 + \alpha} \right), \quad (9)$$

and we use Monte Carlo integration (Bishop, 2006) to approximate the integral. The likelihood from equation 8 also must be approximated for popular motion patterns, as the computations in equation 5 are cubic in the cluster size n_j. Similar to Rasmussen and Williams (2005), we approximate the likelihood for these larger clusters using the N_{max} trajectories that are closest to the trajectory t^i.[8]

The DP concentration hyperparameter α is resampled using standard Gibbs sampling techniques (Rasmussen, 2000). The GP length-scale and variance hyperparameters are more difficult to resample, so we leverage the fact that their posteriors are extremely peaked and instead always set them to their maximum likelihood values (using gradient ascent). In applications where the posteriors are less peaked, hybrid Monte Carlo techniques may be used (Duane et al, 1987).

[7] The inference approach described here is taken from our previous work (Joseph et al, 2010).

[8] We tested the validity of this approximation by comparing approximations in which only the nearest points were used to the true likelihood and found no practical difference when discarding 75% of trajectories for large clusters.

Algorithm 1 Motion Model Inference

```
for sweep = 1 to # of sweeps do
    for each motion pattern b_j do
        Draw the GP hyperparameters θ^GP_{x,j}, θ^GP_{x,j}
    end for
    Draw the DP hyperparameter α
    for each trajectory t^i do
        Draw z_i using equations 8 and 9
    end for
end for
```

3.2.2 Classification and Prediction with New Trajectories

The motion model from algorithm 1 can now be used to predict a target's future locations, given a partial trajectory t^i. We first apply equations 8 and 9 to compute the relative probability of it belonging to each motion pattern b_j. Equation 3 is used to compute the likelihoods. Just as in section 3.2.1 where we trained the model using complete target trajectories, the partial trajectory may contain any number of points. We can use the same equations 8 and 9 to determine the most likely motion patterns for the partial trajectory.

For each likely pattern b_j, we first compute the expected trajectory derivatives $(\frac{\Delta x}{\Delta t}, \frac{\Delta y}{\Delta t})_j$ conditioned on GP parameters $(\theta^{GP}_{x,j}, \theta^{GP}_{y,j})$ (equation 5). The expected trajectory derivative is a weighted average over all the conditional derivatives $\sum_j \pi_j (\frac{\Delta x}{\Delta t}, \frac{\Delta y}{\Delta t})_j$.[9] We apply this expected trajectory derivative to the target's most recent location to predict where it will be in the future.

3.3 Results

In this section we describe our results on two examples. The first is a synthetic single-trajectory scenario where the agent must intercept and track 50 targets, one after the other. The second scenario is a (simulated) helicopter-based tracking scenario in which the targets are cars whose paths are collected from a real dataset. In both cases, we tested our models in an online fashion: initially our agent had no experience with the target; after each episode, the target's full trajectory was incorporated into the motion model.

We compare our DPGP motion model to a Markov model that projects positions and velocities to a discretized grid and uses the trajectory data to learn target transition probabilities between grid cells. The Markov model predicts a target's next grid cell using the transition probabilities stored at the grid cell closest to the target's current position and velocity. In contrast to the Markov model, which ignores trajectory history, the DPGP model considers the entire observed

[9] In practice, we found that the motion pattern likelihoods were highly peaked. In this situation, it was sufficient to only consider the maximum likelihood motion pattern when predicting the future locations in partial trajectories.

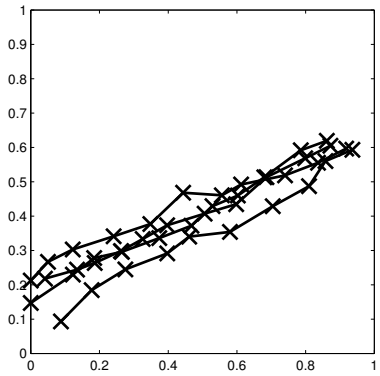


Fig. 7 Several trajectory samples from the CORRIDOR scenario, where targets roughly following a straight line

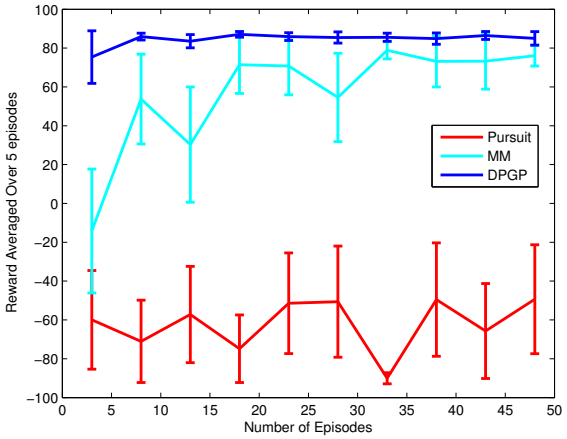
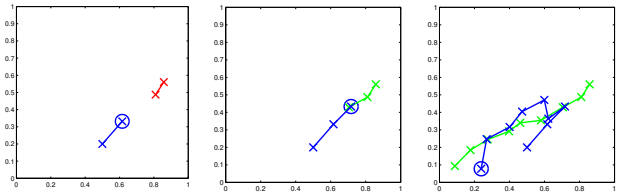


Fig. 8 Sliding window average of per-episode rewards achieved by different models on the CORRIDOR scenario. Error bars show the 95% confidence interval of the mean from five repeated runs.



(a) $t = 2$ (b) $t = 3$ (c) $t = 9$

Fig. 9 A planning episode for a single path in the CORRIDOR scenario. Agent positions are shown in blue and untagged target positions are shown in red (before they are tagged) and green (after they are tagged). The small blue circle and the large cyan circle around the agent signify the tagging and observation range, respectively.

portion of the trajectory when predicting both the target's motion pattern and future trajectory.

3.3.1 Results on a Simple Synthetic Example

We first apply our approach to a simple example involving a target following a straight line with occasional deviations (for example, walking along a puddle-covered road). The agent receives a reward of -10 for every time step until it intercepts the target, whereupon it receives a reward of +100. The agent's task involved intercepting and tracking 50 targets one after the other. We call this the CORRIDOR scenario. Figure 7 shows several trajectories from this example.

Figure 8 shows the results for five repetitions of this set of tasks. For comparison, we plot the results of both the Markov model and a naive pursuit approach that moves the agent to the target's most recent position. Overall, we see that while the agent planning with the Markov models with various initializations eventually reach the same level of performance as the agent using the Gaussian process, the Gaussian process motion model generalizes faster from the data. Figure 9 shows an example planning sequence derived using the Gaussian process motion model in which the agent first intercepts the target and then keeps it in view.

While this is a simple and easy example, we note that the DPGP still outperforms the other models. The DPGP learns the model almost instantaneously, but the Markov model requires approximately 40 trials before matching the performance of the DPGP.

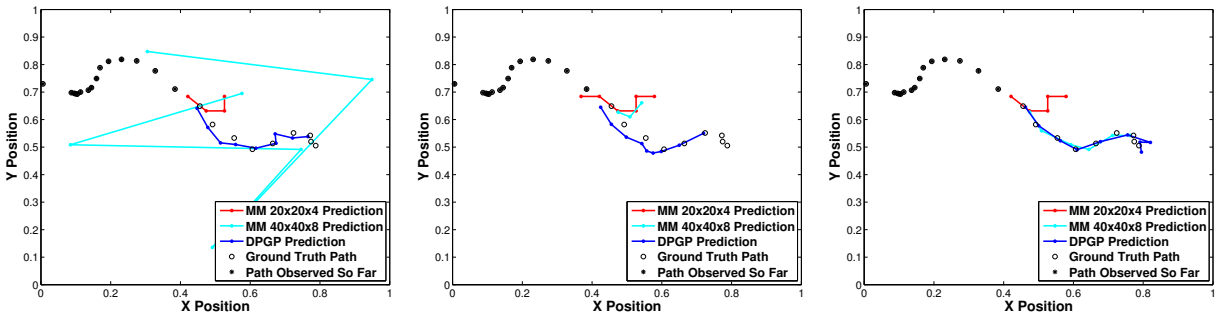
3.3.2 Results on a Helicopter-based Tracking Scenario

Next, we tested our approach on a helicopter-based target-tracking scenario.[10] To model the helicopter and its rewards, we place a 20×20 grid over a city (an area of approximately 10 square miles) and represent the helicopter's state with the closest grid cell. At each time step, the helicopter can stay in place, move one cell, or move two cells. These actions result in rewards of 0, -1, and -2, respectively. The helicopter also receives a reward of 10 for each time step it shares a grid cell with the target car. While a real "chase" scenario would have many more complexities, this simplified tracking task allows us to show empirically that our model, initially trained on likelihood-based criteria, also performs well on a planning problem based on real data.[11]

We tested both our DPGP and the Markov model on 500 trajectories taken from the CarTel dataset of time-stamped GPS coordinates of greater-Boston area taxis. The Markov model was initialized with a uniform prior, and its transition probabilities were updated as new trajectories arrived. To assess the effect of discretization granularity on the Markov model, we evaluated Markov models with different position and velocity resolutions. The x and y-positions were discretized on a 20×20, 40×40, or a 60×60 grid (the helicopter's discretization never changed). Velocity was either

[10] Results in this section are also described in our previous work (Joseph et al, 2010).

[11] Likelihood-based methods try to explain the data well, while the goal of the planning problem is to maximize rewards. A model that best explains the data is not guaranteed to be the best model for planning.



(a) 100 paths (b) 300 paths (c) 500 paths

Fig. 10 Predictions given a partial path for the DPGP and two Markov models for various amounts of training data.

discretized into four or eight states. The models with finer discretizations were more expressive but require more data to train effectively.

After each trajectory was completed, our DPGP driver model was updated using algorithm 1. Each update was initialized with the most recently sampled model. Since a full update required significant computation, new trajectories were initially clustered with their most likely motion pattern (which could have been a new pattern) using equations 8 and 9.

Every 10 new trajectories, a complete set of 5 Gibbs sweeps (algorithm 1) were run to update the model parameters and trajectory assignments (we found that samples generally stopped changing after the first 2 sweeps). The noise parameter σ_n in equation 4 was fit from the current trajectory set. While the DPGP model required more computation than the Markov model (about 10 times slower), it could still incorporate a new set of samples in minutes, an update rate fast enough for a real scenario where the model may be updated several times a day. The planning time was nearly instantaneous for both the DPGP and the Markov driver models.

We first carried out a series of experiments to evaluate the quality of our models. Example predictions of the DPGP and Markov models are seen in figure 10. The solid circles show a partial trajectory; the open circles show the true continuation of the trajectory. The cyan, red, and blue curves show the continuations predicted by the DPGP model and two Markov models. With only 100 training trajectories, none of the models predict the full path, but the DPGP is close while the other models are completely lost. As more training data is added, the DPGP and the finer-grained Markov model match the true trajectory, while the simpler Markov model is not flexible enough to fit the data.

As the goal of our model is to predict the motion of mobile agents within a planner, we compared the performance of planners using the DPGP and Markov models, as well as a naive pursuit approach that simply assumed the vehicle's position at time $t+1$ would be the same as its location at time t. We also evaluated a simple k-nearest neighbor technique

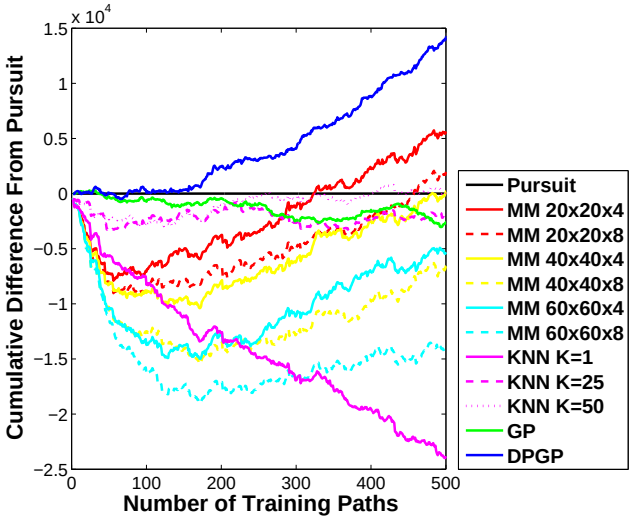


Fig. 11 Cumulative difference in reward from the pursuit approach for the DPGP model, various Markov models (MM), k-nearest neighbors (KNN), and a GP fit to the current trajectory (GP) (higher values are better).

that, given an (x, y) point, simply searched the training set of trajectories for nearby (x, y) points and interpolated the trajectory derivatives $\frac{\Delta x}{\Delta t}$ and $\frac{\Delta y}{\Delta t}$ from the trajectory derivatives of nearby training points.[12] Finally, we evaluated a GP model that was fit to only the current trajectory and ignored all past training data. This single GP model ensured that the previous trajectories were important for making predictions about the current trajectory, that is, the current trajectory could not be well-predicted based on its own velocities.

Figure 11 shows the cumulative difference of total reward between all the approaches and naive pursuit method. The k-nearest neighbor and simple GP rarely out-perform pursuit. The Markov models initially do worse than pursuit because they have many parameters (making them vulner-

[12] For reasonably dense data, Gaussian process and nearest neighbor approximations are very close; thus, the k-nearest neighbor technique also served as a close approximation of a solution trained on a single GP for the entire dataset.

able to over-fitting) and often make incorrect predictions when the agent observes a trajectory in a sparse region of their state space. In contrast, the DPGP starts out similar to pursuit, since the zero-mean prior on trajectory derivatives naturally encodes the bias that, in the absence of other data, the car will likely stay still. The DPGP model quickly generalizes from a few trajectories and thus attains the highest cumulative rewards of the other methods. The Markov models eventually also exhibit similar performance, but they never make up for the initial lost reward.

4 Interception and Tracking with Partial Information

We now consider the case in which the agent does not always have access to the target's current location. Instead, we assume that the agent has a sensor that will provide a perfect measurement of the target's location if the target is within some observation radius of the agent, and no measurement otherwise. The agent's task is to first intercept the target — maneuver to within some small interception radius of the target for "inspection" — and then to keep the target within its larger observation radius. We first formalize the model and detail the inference procedure; we next show how our motion model helps the agent intercept and track targets in a synthetic domain (section 4.3.1) and a helicopter-based search and tracking scenario using the real-world taxi data (section 4.4).

4.1 Interception and Tracking Problem Formulation

Since the target's current position is now potentially unknown at every time step, we formalize the interception and tracking scenario as a partially observable Markov decision process (POMDP). In addition to the states, actions, transition function, and reward function present in an MDP, a POMDP also includes a set of observations and an observation function.

As in the fully observable MDP case (section 3), the state consists of the joint position of our agent and the target $(x^a, y^a, x^{target}, y^{target})$. Given an action and our agent's current position (x_t^a, y_t^a), we assume that our agent's next position (x_{t+1}^a, y_{t+1}^a) is deterministic and known. However, the target's position (x^{target}, y^{target}) is no longer observed. Instead our agent receives an (accurate) observation of the target's position if the target is within an observation radius r_{obs} of our agent. Otherwise our agent receives no information about the target's position. Essentially, we are relaxing the assumption of the previous section that the target is tracked by a dense sensor network. Our agent gets target information at irregular intervals from a sparse sensor network, and must model the target's behavior and plan trajectories to intercept the target given imperfect information about the current target's location. As before, the target's transitions are stochastic over the continuous space; we can place a distribution over the target's next position $(x_{t+1}^{target}, y_{t+1}^{target})$ based on our motion model. The agent receives a large one-time reward for being within a small interception radius of the target (which is significantly smaller than r_{obs} and a small tracking reward for every target within its observation radius.

The inference procedure for learning the target motion models (algorithm 2) is described next in section 4.2; given this model and the remaining problem parameters, the agent chooses actions using a standard forward search (Ross et al, 2008).

Algorithm 2 Partially Observable Motion Model Inference

```
for sweep = 1 to # of sweeps do
    for each trajectory t^i do
        for each time step n do
            if (x_t^target, y_t^target) was not observed then
                Draw (x_t^target, y_t^target) using equation 5
                if (x_t^target, y_t^target) was within r_obs of (x_t^a, y_t^a) then
                    Reject sample, go to 5
                end if
            end if
        end for
    end for
    for each motion pattern b_j do
        Draw the GP hyperparameters θ^GP_{x,j}, θ^GP_{x,j}
    end for
    Draw the DP hyperparameter α
    for each trajectory t^i do
        Draw z_i using equations 8 and 9
    end for
end for
```

4.2 Model Inference

Since our agent sees a target's location only when the target is within a given observation radius, the target trajectory that the agent observes will often be disjoint sections of the target's full trajectory. Fortunately, the Gaussian process does not require continuous trajectories to be trained, and the Dirichlet process mixture model can be used to classify partial paths that contain gaps during which the vehicle was not in sight. In this sense, the inference approach for the full information case (section 3.2) also applies to the partial information case. However, using only the observed locations ignores a key piece of information: whenever the agent does not see the target, it knows that the target is not nearby. In this way, the lack of observations actually provides (negative) information about the target's location.

To leverage this information, we use Gibbs sampling to sample the unobserved target locations as well as the trajectory clusterings. Once the partially observed trajectories are

completed, inference proceeds exactly as in the full information case. Specifically, we alternate resampling the cluster parameters (section 3.2) with resampling the unobserved parts of each target's trajectory. Given all of the other trajectories in an incomplete trajectory's cluster, we can sample the missing sections using the prediction approach in section 3.2.2. If the sampled trajectory crosses a region where the agent could have observed it—but did not—then that sample is rejected, and we sample a new trajectory completion. This rejection-sampling approach ensures that we draw motion patterns consistent with all of the available information (see algorithm 2).

To predict future target positions, several of the sampled trajectory completions are retained and averaged to produce a final prediction. Each trajectory completion suggests a different Gaussian process motion model, and is weighted using Bayesian model-averaging. Using the final velocity prediction, computed as the weighted average of individual model predictions, we can then apply the prediction and classification approach in section 3.2.2 for intercepting and tracking new targets.

4.3 Results

In this section, we apply our DPGP model to two partially observable interception and tracking problems. The first is a synthetic example designed to show the basic qualities of the DPGP in the partially observable case. In the second problem, we return to a more challenging, partially observable version of the taxi tracking scenario from section 3. As in the fully observable case, we tested each model in an online fashion: initially the agent had no experience with the target; after each episode, any information it received about the target was incorporated into the motion model. Specifically, if the agent only observed the target at certain times, only those locations were used to update the motion model. The agent does not receive any additional information about the missed observations of a target's trajectory after an episode.

In all of the scenarios, we compared our DPGP algorithm to a pursuit forward search algorithm, and a Markov model. The pursuit algorithm goes to the target's last observed location but uses forward search to plan about how best to intercept and track all three targets. The Markov models use a position discretization equal to the interception region with x and y velocity each discretized into two bins. It is initialized with a small probability mass on self transitions to encode the bias that in the absence of data the target will tend to stay in the same location. Without this bias the model performs extremely poorly initially. Similar to the other two approaches, the Markov model also uses forward search to plan for the helicopter. While we could have used other Markov models with more bins, the results from section 3.3 show us that these Markov models may perform better in the limit of infinite data but with the small data set here a Markov model with a small number of bins will perform the best.

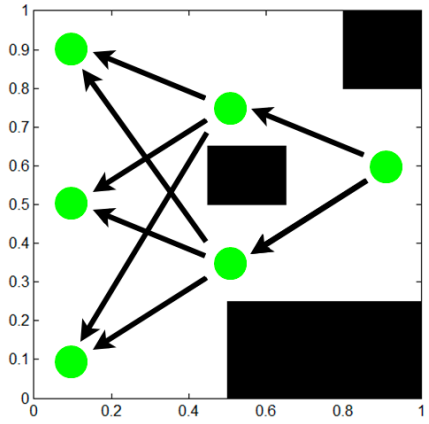


Fig. 12 Search and tracking task in the synthetic BLOCKS scenario.

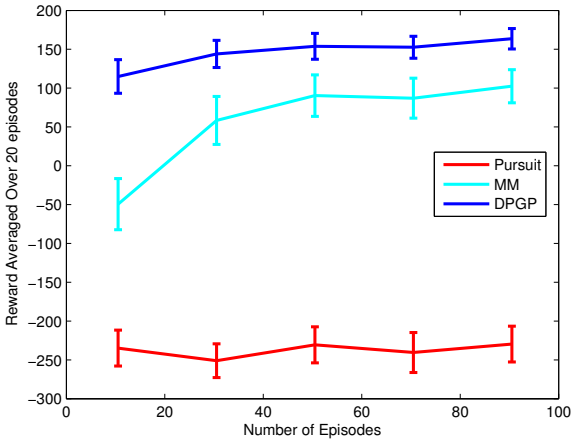
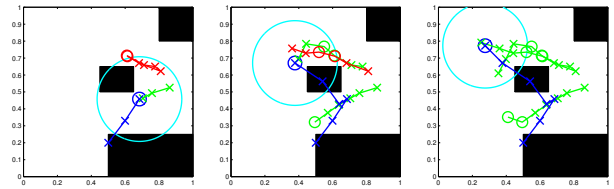


Fig. 13 Sliding window average of per-episode rewards achieved by different models on the BLOCKS scenario. Error bars show the 95% confidence interval of the mean from five repeated runs.

4.3.1 Results on a Synthetic Multi-Target Scenario

We first illustrate our approach on a synthetic interception and tracking problem based on Roy and Earnest (2006). In this problem, illustrated in figure 12, the agent starts near the opening on the far right and must track three targets which start from the right side of the region and simultaneously move to three different target locations on the left wall. Targets have $\frac{3}{4}$ probability of going above the central obstacle and $\frac{1}{4}$ probability of going below it. The agent receives a reward of -10 for every time step until it intercepts the target, whereupon it receives a reward of +100. Additionally, it receives a reward of +1 for every target within its observation radius. We call this the BLOCKS scenario.



(a) $t = 3$ (b) $t = 6$ (c) $t = 7$

Fig. 14 A planning episode in the BLOCKS scenario. Agent positions are shown in blue and target positions are shown in red before they are intercepted and green after. The small blue circle and the large cyan circle around the agent signify the interception region and observation radius, respectively. Target locations that were within the agent's sensor range are marked by $\times$ symbols, and target locations beyond the agent's sensor range are marked with $\circ$ symbols.

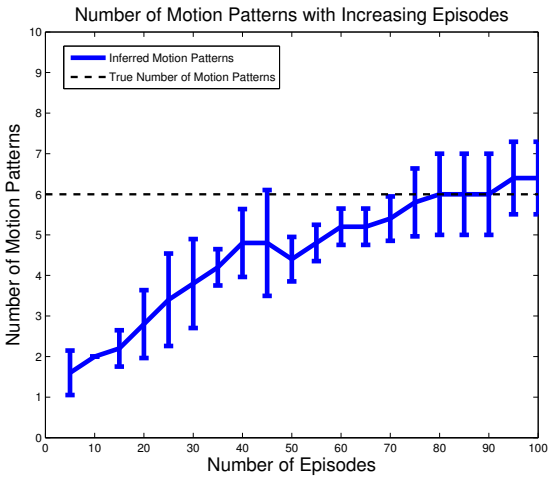


Fig. 15 Number of discovered target motion patterns in the BLOCKS scenario.

Figure 13 shows the performance of each approach over five runs, where each run consists of 100 episodes. The error bars show the 95% confidence interval of the mean from the five runs. This shows that not only are the means of the DPGP approach higher than the other approaches, but in practice it scores significantly better on any particular run. The Markov models, despite requiring a fair amount of data to start making relatively good predictions, do outperform the simpler strategy. Figure 14 shows parts of a single planning episode, where the helicopter initially intercepts one target going below the obstacle before pursuing the last two above the obstacle.

Since this is a synthetic example, we can also compare the motion patterns found to the true underlying patterns in the model. The model has six patterns: the target can go either above or below the obstacle to reach one of the three final locations on the left wall. The number of clusters found by our DPGP approach as a function of training paths is shown in figure 15. In the beginning, when the agent has seen relatively little data, it maintains a smaller number of motion patterns. As the agent observes more trajectories, we see that the number of motion patterns settles around the true number (the error bars show 95% confidence intervals of the mean). By the end of the 100 trials, if two trajectories belonged to the same true cluster, then our DPGP model also placed them in the same cluster with probability 0.7375; if two trajectories actually belonged to separate clusters, then our DPGP model placed them in separate clusters with probability 0.8433. Some of this clustering error is due to our agent being out of range of the target resulting in some trajectories not containing the full location history. In fact, approximately 20% of the data points were not observed during the trails. These statistics, consistent over five runs of the 100 episodes, strongly suggest that our DPGP model was learning key clustering characteristics of the target motion patterns.

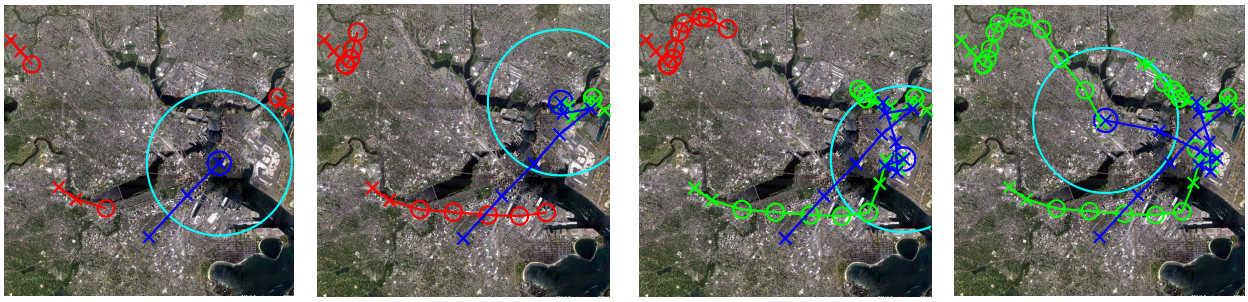
4.4 Results on a Helicopter-based Multi-Target Scenario

We next applied our approach to a helicopter-based search and tracking scenario that used the same taxi dataset described in section 3.3. We assume that the agent was given the targets' true initial locations and velocities from a ground-based alert network. After being given this initial piece of information about the targets, the target states are no longer directly accessible, and the helicopter receives information about a target's location only if the target is within about 1.5 miles ($\frac{1}{4}th$ the map size) of the helicopter. The interception radius is 0.25 miles ($\frac{1}{25}th$ the map size). The reward function is identical to the one described in section 4.3.1.

Figure 16 shows an example episode where the helicopter first intercepts each target and then finds a location where it can observe multiple targets to keep them localized. The results comparing our DPGP approach to the same control strategies from section 4.3.1 are shown in figure 18 and figure 17, with the error bars showing the 95% confidence interval of the mean for the five runs of 100 tasks. Using our DPGP approach for modeling the targets results in much better interception and tracking performance from the start. Unlike the simpler BLOCKS scenario, the Markov models do no better than simple pursuit after 100 episodes. Finally, figure 19 shows the number of clusters found by the DPGP approach as a function of training paths. As expected from a real-world dataset, the number of motion patterns grows with the number of episodes as new motion patterns observed in new trajectories.

5 Discussion

Overall, using our Bayesian nonparametric DPGP approach for modeling target motion patterns improved our agent's



(a) $t = 3$ (b) $t = 7$ (c) $t = 11$ (d) $t = 13$

Fig. 16 A planning episode from the taxi data set. Helicopter positions are shown in blue. Car positions are shown in red before interception and green after. The small blue circle and the large cyan circle around the helicopter signify the tagging and observation range, respectively. Car locations are marked with a $\times$ symbol when observed by the helicopter, and a $\circ$ symbol when beyond the helicopter's sensor range.

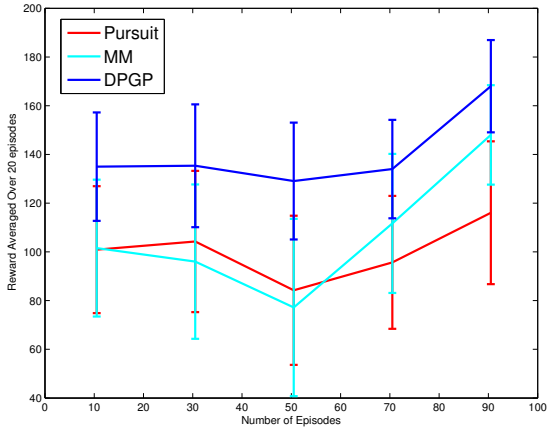


Fig. 17 Sliding window average of per-episode rewards achieved by different models on the taxi multi-target interception and tracking task. Error bars show the 95% confidence interval of the mean from five repeated runs.

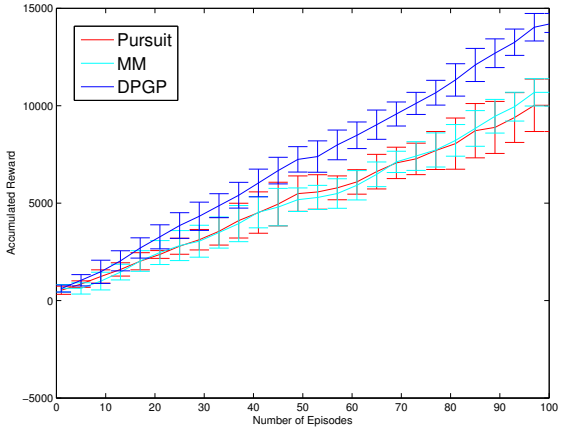


Fig. 18 Results from the taxi multi-target interception and tracking task showing cumulative reward achieved by different models on the BLOCKS scenario. Error bars show the 95% confidence interval of the mean from five repeated runs.

ability to predict a target's future locations from relatively few examples. A key advantage of the DPGP is that it provides a way of scaling the sophistication of its predictions given the complexity of the observed target trajectories: we could model motion patterns directly over a continuous space without needing to specify discretization levels or expected curves. In contrast, the Markov models suffered because even at a "reasonable" discretization, these models needed to train the motion model for every grid cell—which required observing many more trajectories.

While we focused on the motion patterns of taxis in the Boston area, as seen in our synthetic example, the DPGP approach is not limited to modeling motion patterns of cars. It is meant as a far more general mobile agent model, which models a wide variety of trajectories over a continuous space as long as the targets motions obey local smoothness and continuity constraints. The DPGP model is also best suited for situations where complex dynamics and clusterings must be learned from relatively little data—as we saw in the results sections, the Markov models do catch up in performance once sufficient data is available; however, the DPGP makes significantly better predictions from only a few trajectories. Finally, because of the Bayesian nature of our approach, available expert knowledge about target motion patterns could be given in the form of additional example trajectories without any need to adjust the rest of the inference process.

6 Related Work

Much of the past work in modeling mobile agents has focused on two problems: expert systems (which require specialized data) and modeling a single agent (requiring data generated by the single agent). Letchner et al (2006) built a model that predicted trajectories based on the optimal path between two locations (given factors such as the time of day) and the amount of "wasted time" a driver was willing to ac-

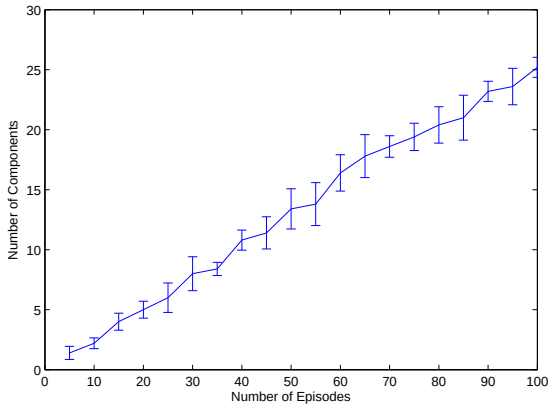


Fig. 19 Number of discovered motion patterns for the taxi dataset search and tracking task.

cept. Dia (2002) used a survey to classify drivers into different profiles to enable better prediction. Both of these works note that it is difficult to specify a model for human motion patterns based on logical reasoning. For example, Letchner et al (2006) note only 34.5% of drivers choose the fastest route between two locations.

Whether these statistics are a result of driver ignorance or another factor (*e.g.*, avoiding a stressful route) is highly debatable and difficult to incorporate into expert models of human motion patterns. Without access to similar data for the greater Boston area or having similar time-stamped GPS data for their models, we were unable to compare them to our approach; however, it would be interesting to see if incorporating expert features related to human psychology into the priors of our model could improve predictions.

Another body of literature has trained Markov models (generally using data from only one person) in which each road segment is a state and transition probabilities encode the probabilities of moving from one segment to another. For example, Patterson et al (2003) treated the true driver state as hidden by GPS sensor noise and a hidden driver mode. Ashbrook and Starner (2003) model the end position and transition probabilities explicitly, but doing so prevents the method from updating the probabilities based on a partially observed trajectory. Using a hierarchy of Markov models, Liao et al (2007) were able to make both local and destination predictions but still had difficulty in regions of sparse training data. Taking a machine learning approach, Ziebart et al (2008) used inverse reinforcement learning with good results when the target's destination is known in advance.

Recently, Gaussian processes have been successfully applied to modeling and prediction in robotics tasks. Tay and Laugier (2007) used a finite mixture of Gaussian processes to model multiple moving targets in a small simulation environment. In the context of controlling a single vehicle, Ko and Fox (2009) demonstrated that Gaussian processes improved the model of a vehicle's dynamics.

Fox et al (2007) took a related approach to ours and modeled the number of motion patterns with a Dirichlet process prior, with each motion pattern governed by a linear-Gaussian state space model. Unlike our approach, agents could switch between motion patterns using an underlying hidden Markov model. In our specific dataset and application, the agents usually know their start and end destinations from the very beginning; not allowing motion pattern changes helped predict a car's path on roadways that were common to many motion patterns. However, our framework could certainly be extended to allow agents to change motion patterns. Future work could also incorporate additional information—such as inputs of the road network—to further constrain the trajectories.

The target-tracking problem under partial observability conditions has a natural formulation as a POMDP, since the agent must make decisions with incomplete knowledge of the targets. Pineau et al (2003) first applied the PBVI point-based solver to a small target-tracking problem, and more recent approximate point-based techniques, for example by Hsu et al (2008) and Kurniawati et al (2009), have expanded the applicability of general POMDP solvers to the target-tracking domain by rapidly exploring the reachable and high-value regions of the belief space. Despite these advances, point-based POMDP methods still have limited utility in this domain. These methods typically discretize the agent and target state spaces to obtain a finite-dimensional belief space, and are unable to adapt to changing motion patterns due to substantial offline requirements.

One approach to avoiding state space discretization is to represent beliefs using Gaussian distributions, as applied by Miller et al (2009) to target tracking, or by He et al (2010) with Gaussian mixture models. An advantage of these representations is the ability to analytically and exactly manipulate the belief state. However, these approaches focus on planning with accurate models, and do not address model learning or acquisition.

7 Conclusion

Accurate agent modeling in large domains often breaks down from over-fitting or under-fitting the training data. We used a Bayesian nonparametric approach to motion-pattern modeling to circumvent these issues. This approach allows us to build flexible models that generalize sensibly with sparse data and add structure as more data is added. The reward models, the dynamics model of the agent, and the form of the agent's planner can all be adapted to the task at hand with few adjustments to the DPGP model or inference procedure.

We demonstrated our motion model on a set of helicopter-based interception and tracking tasks trained and tested on a real dataset of complex car trajectories. The results suggest that our approach will be useful in a variety of agent-modeling situations. Since the underlying structure of our model is based on a Gaussian process framework, our approach could easily be applied to beyond car domains to generic metric spaces. Finally, although we focused our approach on a set of interception and tracking tasks, we note that the DPGP motion model can be applied to any task where predictions about a target's future location are needed.

References

Ashbrook D, Starner T (2003) Using GPS to Learn Significant Locations and Predict Movement Across Multiple Users. Personal Ubiquitous Computing 7(5):275–286

Bishop CM (2006) Pattern Recognition and Machine Learning (Information Science and Statistics). Springer

Deisenroth MP, Huber MF, Hanebeck UD (2009) Analytic Moment-based Gaussian Process Filtering. In: Bouttou L, Littman M (eds) Proceedings of the 26th International Conference on Machine Learning, Omnipress, Montreal, Canada, pp 225–232

Dia H (2002) An Agent-Based Approach to Modeling Driver Route Choice Behaviour Under the Influence of Real-Time Information. The American Statistician 10

Duane S, Kennedy AD, Pendleton BJ, Roweth D (1987) Hybrid Monte Carlo. Physics Letters B 195(2)

Fox E, Sudderth E, Willsky AS (2007) Hierarchical Dirichlet Processes for Tracking Maneuvering Targets. In: Proc. Inter. Conf. Information Fusion, URL `http://www.truststc.org/pubs/260.html`

Girard A, Rasmussen CE, Quintero-Candela J, Murray-smith R (2003) Gaussian process priors with uncertain inputs - application to multiple-step ahead time series forecasting. In: Advances in Neural Information Processing Systems, MIT Press, pp 529–536

He R, Bachrach A, Roy N (2010) Efficient planning under uncertainty for a target-tracking micro-air vehicle. In: Proc. IEEE International Conference on Robotics and Automation

Hsu D, Lee WS, Rong N (2008) A point-based POMDP planner for target tracking. In: Proc. IEEE International Conference on Robotics and Automation, Pasadena, CA

Joseph JM, Doshi-Velez F, Roy N (2010) A bayesian nonparametric approach to modeling mobility patterns. In: AAAI

Ko J, Fox D (2009) GP-BayesFilters: Bayesian filtering using Gaussian process prediction and observation models. Autonomous Robots 27(1):75–90

Kurniawati H, Du Y, Hsu D, Lee W (2009) Motion Planning under Uncertainty for Robotic Tasks with Long Time Horizons. In: Proc. International Symposium of Robotics Research

Letchner J, Krumm J, Horvitz E (2006) Trip Router with Individualized Preferences (TRIP): Incorporating Personalization into Route Planning. In: AAAI

Liao L, Patterson DJ, Fox D, Kautz H (2007) Learning and Inferring Transportation Routines. Artif Intell 171(5-6):311–331

Meeds E, Osindero S (2006) An alternative infinite mixture of Gaussian process experts. In: NIPS 18

Miller SA, Harris ZA, Chong EKP (2009) A POMDP framework for coordinated guidance of autonomous UAVs for multitarget tracking. EURASIP Journal on Advances in Signal Processing

Patterson D, Liao L, Fox D, Kautz H (2003) Inferring High-Level Behavior From Low-Level Sensors. In: Proc. UBICOMP

Pineau J, Gordon G, Thrun S (2003) Point-based value iteration: An anytime algorithm for pomdps. In: International Joint Conference Artificial Intelligence, pp 1025 – 1032

Puterman ML (1994) Markov Decision Processes: Discrete Stochastic Dynamic Programming. Wiley

Raftery A (1986) Choosing models for cross-classifications. Americal Sociological Review 51

Rasmussen CE (2000) The Infinite Gaussian Mixture Model. In: NIPS 12

Rasmussen CE, Ghahramani Z (2002) Infinite mixtures of Gaussian process experts. In: NIPS 14

Rasmussen CE, Williams CKI (2005) Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)

Ross S, Pineau J, Paquet S, Chaib-Draa B (2008) Online planning algorithms for POMDPs. Journal of Artificial Intelligence Research 32:663–704

Roy N, Earnest C (2006) Dynamic action spaces for information gain maximization in search and exploration

Tay C, Laugier C (2007) Modelling smooth paths using gaussian processes. In: International Conference on Field and Service Robotics, URL `http://emotion.inrialpes.fr/bibemotion/2007/TL07`

Teh YW (2007) Dirichlet Processes, submitted to Encyclopedia of Machine Learning

Ziebart BD, Maas A, Bagnell J, Dey AK (2008) Maximum Entropy Inverse Reinforcement Learning. In: AAAI