Human Aware UAS Path Planning in Urban Environments using Nonstationary MDPs

Rakshit Allamaraju, Hassan Kingravi, Allan Axelrod, Girish Chowdhary
Robert Grande, Jonathan P. How, Christopher Crick, Weihua Sheng

***Abstract*—A growing concern with deploying Unmanned Aerial Vehicles (UAVs) in urban environments is the potential violation of human privacy, and the backlash this could entail. Therefore, there is a need for UAV path planning algorithms that minimize the likelihood of invading human privacy. We formulate the problem of human-aware path planning as a nonstationary Markov Decision Process, and provide a novel model-based reinforcement learning solution that leverages Gaussian process clustering. Our algorithm is flexible enough to accommodate changes in human population densities by employing Bayesian nonparametrics, and is real-time computable. The approach is validated experimentally on a large-scale long duration experiment with both simulated and real UAVs.**

I. Introduction

With loosening regulation from the FAA, Unmanned Aerial Systems (UAS) are expected to be deployed for several civilian applications, including package delivery, law enforcement and outdoor monitoring. However, most UAS are equipped with sensors such as cameras, which lead to growing concerns among the public that UAS on missions that require flying over inhabited areas could invade privacy by taking pictures of humans in private locations (see e.g. [1,2]). Authors have discussed whether such pictures taken by UAS would violate the Fourth Amendment [2]. Human-aware UAS path planning algorithms that minimize the likelihood of a UAS flying over areas with high human density could potentially address this issue. Such algorithms would also be useful for covert military applications for conducting tactical missions without being detected, and in household robotics tasks for ensuring that robots do their jobs without interfering in human activity.

In order to create a human-aware UAS path, the UAS must be able to plan in anticipation of where humans could be. A naïve way of planning such paths might take into account available datasets of human population concentration as a function of built-up space, such as census maps maintained by counties or openly available maps such as those available from Google. However, such publicly available data may not be up to date and may not provide sufficient resolution to accurately predict population densities.

Moreover, this approach fails to take into account the fact that human concentration is uncertain, and is often a function of the time of the day or the season. Consider, for example, an urban package delivery UAS. If an *a priori* model of human population density over the urban area is available, then several existing path planning algorithms, including sample-based approaches [3] or Markov Decision Process (MDP) [4] based approaches can be used. Even if such a model is built from historic datasets, or heuristics (such as the fact that highways are busy during rush hour on week-days), it would be difficult to ensure that this model is continually updated to account for unexpected human behavior (such as a game day or construction). Therefore, rather than relying solely on historic datasets, it is beneficial that the UAS maintain a real-time updated model of the population density. Such an model will inherently be non-stationary, so as to account for time-dependent variations. Additionally, human behavior and density patterns often recur. For example, business districts in a city are busier on workdays. Ideally, if the UAS encounters a density pattern that it has seen before, it should be able to reclassify it as a previously-seen model and leverage a policy that it has already computed.

In this paper, we propose a non-stationary Markov Decision Process (MDP) formulation of the human-aware path planning problem, and propose a model-based reinforcement learning approach as a solution to the problem. Our algorithm learns and maintains a separate model for each distinguishable distribution of human density. We use a Gaussian Process (GP) Bayesian Nonparametric (BNP) model to learn the cost associated with being seen by humans. The main benefit of using the GP BNP is that the model grows with the data, and little prior domain knowledge needs to be assumed. A non-Bayesian hypothesis testing based algorithm is used to cluster and classify GP cost models in real-time, and a model-based policy iteration algorithm [5] is used to solve the MDP associated with each reward model. Additionally, we introduce the fog of war concept to drive a variance based exploration-exploitation strategy. The addition of the fog of war function encourages the UAS to explore regions which it has not explored *recently*. This addition is crucial to ensure that the agent is able to detect changes in the environment. The integrated solution architecture proposed in this paper is quite general, and can be applied to other non-stationary planning and control problems, such as path planning in presence of non-stationary obstacles. Furthermore, it contributes to the literature on non-stationary MDPs with unknown switching in the reward or transition models by providing the first GP based BNP non-stationary MDP solution architecture that explicitly handles the trade-off between exploration and exploitation in

Allamraju, Kingravi, Axelrod, and Asst. Prof. Chowdhary are with the Distributed Autonomous Systems Laboratory of Mechanical and Aerospace Eningeering, Grande and How are with LIDS at MIT. Asst. Prof. Crick is with the department of computer science, and Assoc. Prof. Sheng is with Electrical Engineering at Oklahoma State University, Stillwater OK girish.chowdhary@okstate.edu

dynamically evolving environments.

II. Related Work

The problem of robot task planning in human contexts has begun to be explored in the human-robot interaction (HRI) literature. Early attempts to deploy robots into public spaces such as museums [6] demonstrated the need to include human behavior in path planning. Since then, research has tended to focus on robot behavior aimed at increasing human comfort level [7,8], taking advantage of human social behavior [9,10], improving navigation in crowded environments [11], and increasing predictability of a robot's motion [12]. In our work, we consider the privacy aspect of HRI explicitly, as opposed to past approaches that fold privacy concerns into humans' general comfort levels.

Gaussian processes have previously been used successfully in model-based reinforcement learning and approximate dynamic programming [13,14]. The main attraction of these approaches has been the flexibility afforded by the GPs, and the fact that the GP predictive variance can be used to guide exploration of the domain to areas where the model has little predictive confidence [15]. However, existing work on GP-based planning, and other model-based reinforcement learning work [16,17], has focused on stationary domains, where the reward and the transition models are not time-varying. It is difficult to directly extend existing work to non-stationary domains, because the traditional GP inference algorithms assume stationary generative distributions [18]. Even efficient online versions of existing GP inference algorithms update the predictive variance independent of the measurements, hence the predictive variance can decrease even when the underlying generative model is changing [19]. Pérez-Cruz et al. [20] and Chowdhary et al. [21] have recently proposed adding time to the Gaussian kernel as a way to handle time variations. These approaches result in a forgetting factor which allows for relearning, however determining this forgetting rate *a priori* is difficult in general. More seriously, adding time to the Gaussian kernel prevents model convergence and does not allow for long-term learning. In our application, exploration is expensive and results in more human sightings. Ideally, we would like to learn a model to limit the amount of exploration needed to accurately represent a model. To overcome these limitations, we use a non-Bayesian GP clustering algorithm [22] that detects changes in the underlying generative model and is capable of detecting when a previously-seen model appears again. In combination with the fog of war concept to ensure sufficient and controllable exploration in an exploration-exploitation framework. Nonstationary MDPs have been previously explored, but the focus has either been on regret bounds [23], or on heuristics to account for the nonstationarity [24]. On the other hand, our algorithm learns to leverage repeating patterns to improve performance.

III. Human-Aware Path planning using Nonstationary MDPs

A stationary Markov decision process is a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{T}, r)$, where $\mathcal{S}$ is a state space, $\mathcal{A}$ is an action space, $\mathcal{T} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathcal{S}$ is a transition model, and $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function. Each of these quantities are fixed, and not time varying. The goal of an agent in an MDP setting is to maximize the expected sum of accumulated rewards over a discounted infinite horizon, and the solution can be described by a *policy* $\pi : \mathcal{S} \to \mathcal{A}$, which gives a strategy for actions to take at states.

$$\sum_{t=0} \gamma^t r(s_t, a_t), \tag{1}$$

where $a_t = \pi(s_t)$, and $0 < \gamma < 1$. These discounted state and action *value functions* (i.e. the value of a given state over the infinite horizon, or the value of following a given action over the same) for a given policy π are denoted by V^π and Q^π. If the reward and transition models are *fixed* and known, MDPs can be solved using the Bellman equations [25]. If the reward $r(t)$ is not known fully, the MDP problem can be solved using reinforcement learning approaches [26,27]. In model-based reinforcement learning, a model $\hat{r}(t)$ for $r(t)$ needs to be learned. Once the model is learned, the policy π can be solved for using the tuple $(\mathcal{S}, \mathcal{A}, \mathcal{T}, \hat{r})$. The model-based approach is chosen here because it can be more sample efficient [28], and it has the added advantage that the learned model of the environment can be used for other purposes.

The inference of r can be thought of as a function approximation problem. Consider the space of functions $\mathcal{H}$ s.t. $r \in \mathcal{H}$ for all admissible reward functions r. In this case, r is just a point in $\mathcal{H}$. Now consider a curve in $\mathcal{H}$ i.e. a mapping $\tau : [0, \infty) \to \mathcal{H}$, whose image r_τ is the curve in question. Unlike the standard definition of a curve, we do not assume τ to be continuous, i.e. there can be a switch between two functions at any given instant. A *nonstationary MDP* is an MDP where the reward r_τ arises from such a curve. That is, the reward function can be time varying. To solve a nonstationary MDP in a model-based manner, it is necessary that a model of r_τ be learned. If $\mathcal{H}$ is an arbitrary space of functions, inference of r_τ becomes difficult. However, if we assume $\mathcal{H}$ is a reproducing kernel Hilbert space (RKHS), we can perform simpler inference using Gaussian process regression, which we introduce in the next section. Another major advantage of the GP approach is that it accommodates stochasticity in the model built in form of a mean, measurement variance, and a predictive variance. Furthermore, the predictive variance can be interpreted as a measure of confidence in the model estimate at a given point. This can be used to create a strategy for both exploration and exploitation for the solver, as explained in the sequel.

In this paper, the human-aware path planning problem is formulated as a nonstationary model based reinforcement learning problem. The reason behind this choice is that it allows us to find a general solution in presence of a-priori unknown and time-varying human population densities. We

assume a discrete switching model as opposed to a smoothly changing model of human population densities. This choice is justified because UAS missions are rarely continuous; hence due to arbitrary pauses in-between missions, the population density models will change significantly in a discrete fashion. Furthermore, within a mission cycle, we assume that the human density is approximately the same, although our algorithm can support continuous changes as well as handle stochastic variations. The probability of being seen by a human is modeled as a Gaussian distribution centered around the humans, and the agent receives a stochastic cost between -1 and 0 drawn from this distribution. The net effect is that the agent is highly likely to receive large negative reward when it gets close to human populations. The choice of Gaussian distribution is reasonable when the human and aircraft are within direct line-of-sight. For other situations, a Gaussian link-model type approach can be adopted [29]. The agent receives a positive reward of 1 at the goal location. The objective of the agents is to plan a path that maximizes the total cumulative reward. Such a path would avoid human densities and take the agent to the goal. Additional constraints such as time or fuel can be added, but are avoided here for simplicity. The agent transitions are assumed to be defined over a grid on the domain, and the agent has the choice of four actions over each grid location: forward, backward, right, or left. The resulting policy is then further discretized to yield a set of waypoints that describe the UAV's desired path.

IV. Gaussian Process Regression

A GP is defined as a collection of random variables such that every finite subset is jointly Gaussian. The joint Gaussian condition means that GPs are completely characterized by their second order statistics [30]. A GP is a distribution over functions, that is, a draw from a GP is a function. When a process Δ follows a Gaussian process model, then

$$\Delta(\cdot) \sim \mathcal{GP}(m(\cdot), k(\cdot,\cdot)), \tag{2}$$

where $m(\cdot)$ is the mean function, and $k(\cdot,\cdot)$ is a real-valued, positive definite covariance kernel function. Under GP regression, the mean is assumed to lie in the class of functions $\mathcal{H}$, a reproducing kernel Hilbert space (RKHS).

Let $Z_\tau = \{z_1, \ldots, z_\tau\}$ be a set of state measurements, discretely sampled where $\{1 \ldots \tau\}$ are indices for the discrete sample times $\{t_1, \ldots, t_\tau\}$. The set defines a covariance matrix $K_{ij} := k(z_i, z_j)$. Given indexed sets A and B, $K(A,B)$ denotes the kernel matrix generated by the evaluations $K_{ij} = k(a_i, b_j)$ between the two sets, where $a_i \in A, b_j \in B$. For each measurement z_i, there is an observed output $y(z_i) = m(z_i) + \epsilon_i$, where $\epsilon_i \sim \mathcal{N}(0, \omega^2)$. The stacked outputs give $y = [y_1, \ldots, y_\tau]^T$. The most common choice of covariance kernel, and the one used here, is the Gaussian RBF kernel. It can be shown that the mean function and posterior variance in GP regression can be computed as functions of the covariance matrix, which can be written as a $\tau \times \tau$ Gram matrix [30].

Since both Z_τ and y_τ grow with data, computing the inverse becomes computationally intractable over time. This is less of a problem for traditional GP regression applications, which often involve finite learning samples and offline learning. However, in an online setting, the linear growth in the sample set cardinality degrades computational performance. Therefore, the extension of GP regression for control requires an online method to restrict the number of data points stored for inference. Since the set Z generates a family of functions $\mathcal{F}_Z \subset \mathcal{H}$ whose richness characterizes the quality of the posterior inference, a natural and simple way to determine whether to add a new point to the subspace is to check how well it is approximated by the elements in Z, using the kernel linear independence test [19]. This restricted set of selected elements, called the *basis vector set*, is denoted by $\mathcal{BV}$. When incorporating a new data point into the GP model, the inverse kernel matrix can be recomputed with a rank-1 update. When the budget is exceeded, a basis vector element must be removed prior to adding another element [31]. There are many schemes to remove the basis vector; in our experiments, we rely on a method that efficiently approximates the KL divergence between the current GP and the $(t+1)$ alternative GPs missing one data point each, then deletes removes the data point with the largest KL divergence. See [19] for more details. Note that for the purposes of this paper, different GP models correspond to models trained on different data; models with different hyperparameters are not considered.

V. GP Clustering

The inference method in the previous section assumes that the data arises from a single model. In many real-world applications however, this is not a valid assumption. In particular, an algorithm that can determine that the current model doesn't match the source of data can be very useful, especially in the context of nonstationary reward inference. An algorithm for clustering GP models using a non-Bayesian hypothesis test is presented here based on the GP-NBC algorithm in [22]. The main benefit of GP-NBC here that it can be computationally more efficient that Dirichlet Process GP clustering algorithms since sampling based inference does not need to be performed [32].

For a GP, the log likelihood of a subset of points y can be evaluated as

$$\log P(y_\tau \mid z_\tau, M) = -\tfrac{1}{2}(y_\tau - \mu(z_\tau))^T \Sigma_{z_\tau z_\tau}(y_\tau - \mu(z_\tau)) - \log|\Sigma_{z_\tau z_\tau}|^{1/2} + C, \tag{3}$$

where $\mu(z_{\tau+1}) = K(Z_\tau, z_{\tau+1})^T (K(Z_\tau, Z_\tau) + \omega_n^2 I)^{-1} Y$ is the mean prediction of the model M and $\Sigma_{z_{\tau+1}} = K(z_{\tau+1}, z_{\tau+1}) + \omega_n^2 I - K(Z_\tau, z_{\tau+1})^T (K(Z_\tau, Z_\tau) + \omega_n^2 I)^{-1} K(Z_\tau, z_{\tau+1})$ is the conditional variance plus the measurement noise. The log-likelihood contains two terms which account for the deviation of points from the mean, $\frac{1}{2}(y_\tau - \mu(z_\tau))^T \Sigma_{z_\tau z_\tau}(y_\tau - \mu(z_\tau))$, as well as the relative certainty in the prediction of the mean at those points $\log|\Sigma_{z_\tau z_\tau}|^{1/2}$. Our algorithm, at all times, maintains a set of points S which are considered unlikely to have arisen from the current GP model M_c. The set S is used to create a new GP M_S, which is tested against the existing models M_i using a non-Bayesian hypothesis test to determine whether the new

Algorithm 1 GP Clustering

Input: Initial data (Z, Y), lps size l, model deviation η
Initialize GP Model 1 from (Z, Y).
Initialize set of least probable points $S = \emptyset$.
while new data is available **do**
  Denote the current model by M_c.
  If data is unlikely with respect to M_c, include it in S.
  if $|S| == l$ **then**
    for each model M_i **do**
      Calculate log-likelihood of data points S using having been generated from current model M_i (3) $\log(S|M_i)$, and find highest likelihood model M_h, making M_h current model.
      Create new GP M_S from S.
      if $\frac{1}{l}(\log(S|M_S) - \log(S|M_c)) > \eta$ **then**
        Add M_S as a new model.
      end if
    end for
  end if
end while

model M_S merits instantiation as a new model. This test is defined as

$$\frac{P(y \mid z, M_i)}{P(y \mid z, M_j)} \underset{\hat{M}_j}{\overset{\hat{M}_i}{\gtrless}} \eta \tag{4}$$

where $\eta = (1-p)/p$, and $p = P(M_1)$. If the quantity on the left hand side is greater than η, then the hypothesis M_i (i.e. that the data y is better represented by M_i) is chosen, and vice versa. The overall algorithm is described in Algorithm 1; see [22] for more details.

VI. Proposed Solution to Nonstationary MDPs

In this section, we leverage the above ideas to create an algorithm for human-aware path planning and more generally for nonstationary MDPs with unknown reward models. In the algorithm, the GP clustering algorithm builds the model estimate $\hat{r}_{t_i}$ for the reward r_{t_i}, at a given moment in time t_i online. Based on the current model of the reward, a decision is made to either a) explore the state space to gather more information for the reward, or b) exploit the model by solving the MDP $(\mathcal{S}, \mathcal{A}, \mathcal{T}, \hat{r}_{t_i})$. While step b) is clear, the question arises on *how* to efficiently explore $\mathcal{S}$ so as to minimize time spent following a potentially suboptimal policy.

A. GP Exploration

Recall that in GP inference, the predictive variance is an indication of the GP's confidence in its estimate at a given location. Therefore, in areas that have been unexplored, the predictive variance is high and vice versa. This observation gives us an indication of how the GP's own estimate of the world can be used as a tool for exploration. We define the *exploration reward* $\hat{r}_{e_{t_i}}$ at time t_i as

$$\hat{r}_{e_{t_i}}(x) = \hat{\Sigma}_{t_i}(x) + \Upsilon_{t_i}(x), \tag{5}$$

Algorithm 2 Nonstationary MDP Solver

Initialize: Initial data (X, Y), lps size l, model deviation η, GP parameters (σ, ω_n^2), space exploration threshold φ.
while new data (x_i, y_i) is available **do**
  Update GP cluster using Algorithm 1.
  Compute exploration reward $\hat{r}_{e_{t_i}}$ using (5).
  Compute space explored s_e using (6).
  if $s_e > \varphi$ **then**
    Solve exploitation MDP $(\mathcal{S}, \mathcal{A}, \mathcal{T}, \hat{r}_{t_i})$ to get path.
  else
    Solve exploration MDP $(\mathcal{S}, \mathcal{A}, \mathcal{T}, \hat{r}_{e_{t_i}})$ to get path.
  end if
end while

where $\hat{\Sigma}_{t_i}(x)$ is the covariance of the GP over $\mathcal{S}$, and Υ_{t_i} is a functional over $\mathcal{H}$ which we will refer to as the *fog of war* functional. The latter increases the predictive variance as a function of time, so as to induce a tunable forgetting factor into (5). This functional is defined in this manner for maximal generality; a very simple choice however, and the one we use in this paper is $\Upsilon_{t_i}(x) := \max(C_{fow_1}(t - t_s), C_{fow_2})$, where t_s is the last time the model switched, and $C_{fow_1}, C_{fow_2} \in \mathbb{R}_+$ are user-defined constants. This exploration reward is used to create a new *exploration MDP* $(\mathcal{S}, \mathcal{A}, \mathcal{T}, \hat{r}_{e_{t_i}})$, which can be solved in order to explore $\mathcal{S}$. With this intuition in place, we can have a natural rule to make a decision on when to explore, by computing the quantity

$$s_e(t_i) = \kappa - \frac{1}{\mathrm{vol}(D)} \int_D \hat{r}_{e_{t_i}}, \tag{6}$$

where κ is the largest value of the variance (1 for the Gaussian kernel) and $D \subset \mathcal{S}$. The quantity s_e is a measure of the space explored at an instant t_i. If this is above a certain threshold, the agent should only exploit its knowledge of $\mathcal{S}$ instead of exploring. The use of predictive variance for exploration is similar to "knownness" based model-free MDP solvers [33] and information entropy maximizing active sensor placement algorithm [15]. The idea is extended here to nonstationary MDPs. In fact, it should be noted that the above idea is best used when GP clustering is employed, and is less effective in the case of simply using one GP. This is due to the fact that the computation of the predictive variance depends only on the states visited and not the observations y_i. Therefore, if a model switch occurs, a model utilizing a single GP has to solely rely on $\Upsilon_{t_i}(x)$ to dictate when to explore, because the predictive variance will be low for the model. These effects are visible in the experiments in Section VII. Putting it all together, we get the algorithm shown in Algorithm 2.

VII. Experimental Results

A. Description of the Experiment

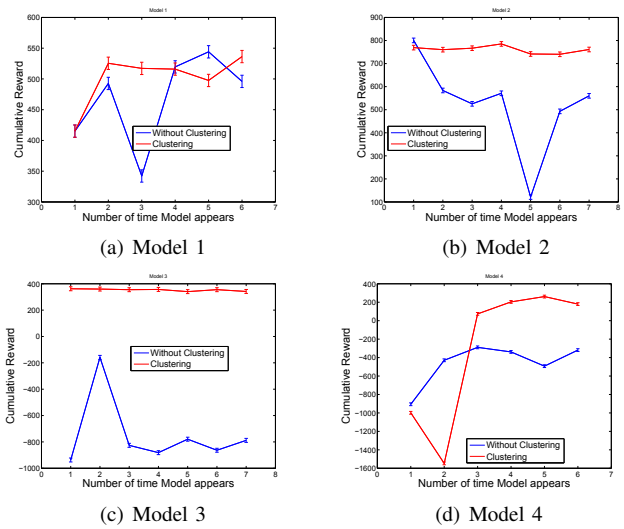
The goal of the experiment is to validate the performance of the proposed planner. The mission scenario is for the agents to go from a predefined ingress location to a predefined egress location (referred to as the goal location) on the domain. The

agent is free to choose the optimal path over the domain to perform its mission. Each path planning and execution instance is termed as a run. A large-scale experiment with a total of $5,500$ runs across the environment is performed. During the experiment, the agents face four different population densities. The population densities switch periodically every 200 runs for the first 4 times for each model, and then randomly. The agents do not have an a-priori model of the underlying human population densities, nor do they know that these are expected to switch or know the total number of underlying models. Furthermore, stochasticity in transitions is induced by any tracking error the Quadrotor controller may have. The rewards samples are stochastic draws from the corresponding GP.

The urban arena is split into grids of dimension 50×50, and the agent action set consists of a decision to move to any directly adjacent grid. Both the GP Clustering (GPC)-based planner and the single-GP Regression (GPR)-based planner process each reward and state location to build a GP based generative model of the reward. A highly efficient policy-iteration based planner [34] is used to compute the (nearly) optimal policy at the beginning of each run using the learned reward model. The resulting policy is further discretized into a set of waypoints, which are then sent to the agent as the desired path. To accommodate a large-number of runs $(5,500)$, simulated UAV agents are used in the experiment. The first four times the population density model is switched, 15 runs are performed by the real-UAV. Hence, the real UAV performs a total of 75 runs across the domain, with each run taking around 2 minutes from take-off to landing. Furthermore, since these runs are early in the learning phase of each model, the paths flown by the UAV are exploratory, and hence longer. The data collected by each agent is assumed to be fed to a centralized non-stationary MDP based planner, which does not distinguish between real and simulated agents. The simulated agents use physics based dynamics model for a medium sized Quadrotor and quaternion based trajectory tracking controllers [35,36]. The real-UAVs fly in an experimental testbed; the testbed area is 16×12 ft and it is equipped with the Optitrack motion capture system and is designed to conduct real-time testing in an emulated urban-environment. The testbed consists of wooden buildings and a couple of cars(all cars not shown in figure) that can move around, allowing us to simulate different population densities. The testbed further includes a monitoring facility with a Mobotix Q24 fish eye camera. The UAV used in this experiment is the a AR-Drone Parrot 2.0 Quadrotor, which is controlled by a client program using the Robot Operating System (ROS) [37] and our own PI controllers.

B. Discussion of Results

Figure 1 presents the average reward accumulated and its variance by both the planners for each of the four models. The horizontal axis indicates the number of times each model is active and the vertical axis indicates the cumulative reward the agent has accumulated over the time the underlying model was active. In model 1, both algorithms perform very similarly. One reason for this is because GPR tends to learn model 1



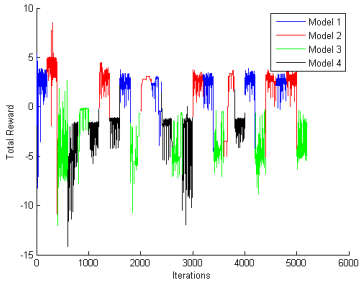
(a) Model 1 (b) Model 2

(c) Model 3 (d) Model 4

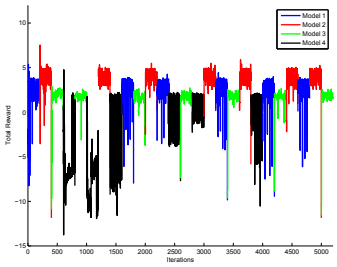
Fig. 1. Comparison between Total Rewards for each model

quickly as this is the model it was initialized in. In fact, the GPR predictive co-variance reduces heavily while learning this model for the first time, and even with the fog-of-war forgetting, the co-variance does not increase again. This results in the GPR (falsely) being confident in its model, and even though it is updated online with new reward samples, its estimate is highly biased towards the first model. However for models 2, 3 and 4, our clustering base GPC algorithm is clearly seen to have a better performance characterized by the consistency over which the GPC based planner finds the near optimal policy. Indeed it can be observed by noticing that total reward accumulated over the entire duration are nearly constant, with the small variations being attributed to the stochasticity in the reward. Figure 2 indicates total reward accumulated by the agent in each iteration, when the underlying reward model is changing. Figure 3 shows the performance of the GPC algorithm. It can be seen that the algorithm rapidly learns the underlying model, and quickly identifies whether the underlying model is similar to a one it has learned before. As a result, the agent's estimate of the reward model converges quickly for each model. Furthermore, because the agent can recognize a previous model that it has already learned before, it does not have to spend a lot of time exploring. The net effect is that the agent obtains consistently high reward, and a long-term improvement in performance can be seen. Whereas, when the agent follows the non clustering based traditional GPR approach, it takes a longer time to find the optimal policy leading to a depreciation in its total accumulated reward over each model.

Some comments on the way exploration is handled in our algorithm and its effect on the long-term performance are in order here. Exploration of the domain is required by any reinforcement learning algorithm to ensure that it obtains sufficient information to compute the optimal policy [27]. In our architecture, the exploration strategy is to compute a policy that guides the agents to areas of the domain where it has little



(a) Cumulative Reward without Clustering



(b) Cumulative Reward with Clustering

Fig. 2. Comparison of the accumulated rewards of GP clustering versus GPRegression; the agent accumulates more positive rewards when clustering models.

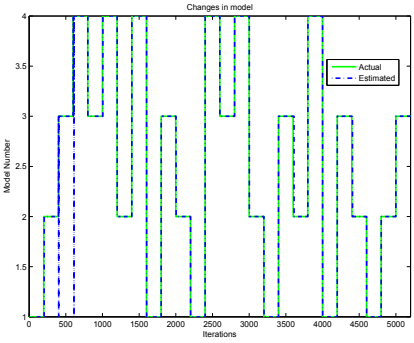
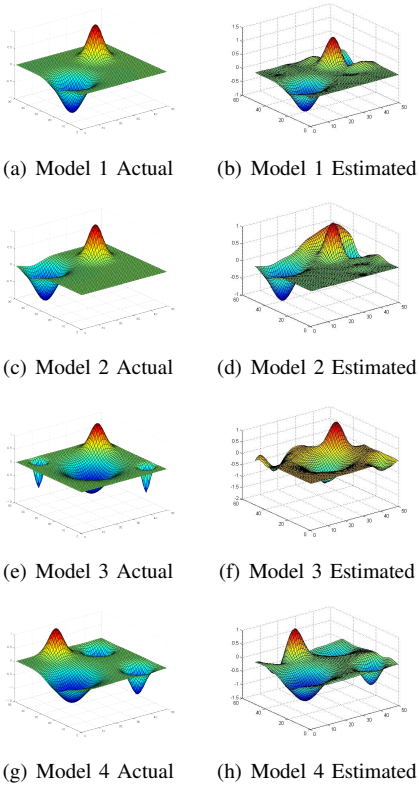


Fig. 3. Plot indicating the actual model being tracked by the estimated model. At the 200^{th} and 400^{th} run new models are introduced, the algorithm quickly detects them after a brief misclassification. Note that the algorithm clusters the underlying reward model quickly and reliably afterwards.

confidence in its model. However, as discussed in Section VI-A, relying simply on the GP variance is not enough, because the GP variance updates in closed-form GP algorithms ([18,19]) do not take into account the non-stationarity in the domain. The fog of war forgetting introduced in (5) adds another metric on learned model confidence by ensuring that the agent revisits parts of the domain that it has not recently visited. Yet, exploration is costly, because this is when the agent is likely to accumulate negative reward. Therefore, the optimal long-term strategy should be to minimize exploration and maximize exploitation by identifying similarities in the underlying models. It becomes clear therefore that the performance of our GPC-based planner is superior because the GPC algorithm is able to cluster the underlying reward model with ones that it has seen before, and hence does not need



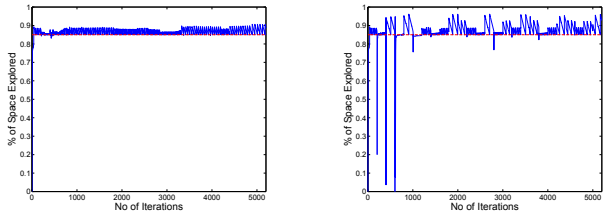
(a) Model 1 Actual (b) Model 1 Estimated

(c) Model 2 Actual (d) Model 2 Estimated

(e) Model 3 Actual (f) Model 3 Estimated

(g) Model 4 Actual (h) Model 4 Estimated

Fig. 4. Estimated and actual mean of a Gaussian Process reward generative model.



(a) Space Explored GPR (b) Space Explored GPC

Fig. 5. Space Explored by each planner indicative of the variance

to explore as much. The estimation performance of the GPC algorithm is visible by comparing the real and estimated models in Figures 4. This indicates that the algorithm did not spend time exploring areas where it cannot perform optimally. Furthermore, Figure 5 plots the value of the exploration reward (6) and the exploitation threshold (0.85). From this figure it can be seen that the GPC planner spends less time exploring. In fact, the dips in Figure 5 match the times when the agent encounters a new model for the first time. These strong dips are desirable, because they indicate that the algorithm has detected that a new model has been encountered, and that it is likely to sufficiently explore the domain to learn that model. In contrast, only one such strong dip is seen at the beginning for the GPR based planner.

VIII. Conclusion

We presented a flexible and adaptive architecture to plan UAV paths to minimize the likelihood of the UAV being seen by humans. Our approach handles the potential shift in

human population densities during the day, season, or time of the year by formulating the planning problem as a non-stationary MDP with unknown and switching reward models, and provides a model-based reinforcement learning solution to the problem. In our architecture, the different underlying reward models generated by the likelihood of being seen by humans are learned using an efficient Gaussian Process clustering algorithm. Sufficient exploration to make clustering decisions was induced through a novel *fog of war* factor that encourages re-visiting of areas not recently visited. The learned reward models are then used in conjunction with a policy iteration algorithm to solve the nonstationary MDP. The proposed architecture is validated in a long-duration experiment with over 5500 simulated and real UAV path planning instances. The results show that the ability of the GP Clustering based planner to learn and recognize previously seen population densities allows our architecture to minimize unnecesary exploration and improve performance over the long term over a tradtional GP regression based approach. Although developed in context of human aware path planning, our architecture is quiet general, and it can be extended to solving other non-stationary MDPs as well.

REFERENCES

[1] P. Finn, "Domestic use of aerial drones by law enforcement likely to prompt privacy debate," *Washington Post*, vol. 22, 2011.

[2] T. Takahashi, "Drones and privacy," 2012.

[3] S. Karaman, M. Walter, A. Perez, E. Frazzoli, and S. Teller, "Anytime motion planning using the RRT*," in *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1478–1483, IEEE, 2011.

[4] S. Thrun, W. Burgard, D. Fox, *et al.*, *Probabilistic robotics*, vol. 1. MIT press Cambridge, 2005.

[5] M. G. Lagoudakis and R. Parr, "Least-squares policy iteration," *J. Mach. Learn. Res.*, vol. 4, pp. 1107–1149, December 2003.

[6] W. Burgard, A. B. Cremers, D. Fox, D. Hahnel, G. Lakemeyer, D. Schulz, W. Steiner, and S. Thrun, "Experiences with an interactive museum tour guide robot," *Artificial Intelligence*, vol. 114, pp. 3–55, 1999.

[7] K. Dautenhahn, M. Walters, S. Woods, K. L. Koay, C. L. Nehaniv, A. Sisbot, R. Alami, and T. Simeon, "How may i serve you? a robot companion approaching a seated person in a helping context," in *Proceedings of the 1st ACM SIGCHI/SIGART Conference on Human-Robot Interaction (HRI)*, 2006.

[8] D. Feil-Seifer and M. Mataric, "People-aware navigation for goal-oriented behavior involving a human partner," in *Proceedings of the IEEE International Conference on Development and Learning (ICDL)*, 2011.

[9] P. Trautman and A. Krause, "Unfreezing the robot: Navigation in dense, interacting crowds," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2010.

[10] D. Althoff, J. J. Kuffner, D. Wollherr, and M. Buss, "Safety assessment of robot trajectories for navigation in uncertain and dynamic environments," *Autonomous Robots*, vol. 32, pp. 285–302, 2012.

[11] T. Bandyopadhyay, K. S. Won, E. Frazzoli, D. Hsu, W. S. Lee, and D. Rus, "Intention-aware motion planning," in *Algorithmic Foundations of Robotics X*, pp. 475–491, Springer, 2013.

[12] C. Lichtenthaler and T. Lorenz, "Influence of legibility on perceived safety in a virtual human-robot path crossing task," in *21st IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*, 2012.

[13] M. Deisenroth, J. Peters, and C. Rasmussen, "Approximate dynamic programming with Gaussian processes," in *Proceedings of the American Control Conference*, 2008.

[14] C. Rasmussen and M. Kuss, "Gaussian processes in reinforcement learning," *Advances in Neural Information Processing Systems*, vol. 16, pp. 751–759, 2004.

[15] A. Krause, A. Singh, and C. Guestrin, "Near-optimal sensor placements in gaussian processes: Theory, efficient algorithms and empirical studies," *Journal of Machine Learning Research (JMLR)*, vol. 9, pp. 235–284, June 2008.

[16] N. K. Ure, A. Geramifard, G. Chowdhary, and J. P. How, "Adaptive Planning for Markov Decision Processes with Uncertain Transition Models via Incremental Feature Dependency Discovery," in *European Conference on Machine Learning (ECML)*, 2012.

[17] N. K. Ure, G. Chowdhary, Y. F. Chen, J. P. How, and J. Vian, "Health-aware decentralized planning and learning for large-scale multiagent missions," in *Conference on Guidance Navigation and Control*, (Washington DC), AIAA, August 2013.

[18] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning*. The MIT Press, December 2005.

[19] L. Csató and M. Opper, "Sparse on-line Gaussian processes," *Neural Computation*, vol. 14, no. 3, pp. 641–668, 2002.

[20] F. Pérez-Cruz, S. Van Vaerenbergh, J. J. Murillo-Fuentes, M. Lázaro-Gredilla, and I. Santamaria, "Gaussian processes for nonlinear signal processing," *arXiv preprint arXiv:1303.2823*, 2013.

[21] G. Chowdhary, H. Kingravi, J. P. How, and P. Vela, "Bayesian nonparametric adaptive control of time varying systems using Gaussian processes," in *American Control Conference (ACC)*, IEEE, 2013 (submitted).

[22] R. Grande, "Computationally efficient gaussian process changepoint detection and regression."

[23] J. Y. Yu and S. Mannor, "Online learning in markov decision processes with arbitrarily changing rewards and transitions," in *Game Theory for Networks, 2009. GameNets' 09. International Conference on*, pp. 314–322, IEEE, 2009.

[24] M. Abdoos, N. Mozayani, and A. L. Bazzan, "Traffic light control in non-stationary environments based on multi agent q-learning," in *Intelligent Transportation Systems (ITSC), 2011 14th International IEEE Conference on*, pp. 1580–1585, IEEE, 2011.

[25] D. Bertsekas, *Dynamic Programming and Optimal Control*. Athena Scientific, 2005.

[26] R. Sutton and A. Barto, *Reinforcement learning: An introduction*. MIT Press, 1998.

[27] L. Busoniu, R. Babuska, B. D. Schutter, and D. Ernst, *Reinforcement Learning and Dynamic Programming Using Function Approximators*. CRC Press, 2010.

[28] C. G. Atkeson and J. C. Santamaria, "A comparison of direct and model-based reinforcement learning," in *IN INTERNATIONAL CONFERENCE ON ROBOTICS AND AUTOMATION*, pp. 3557–3564, IEEE Press, 1997.

[29] Y. Mostofi, A. G. Ruiz, A. Ghaffarkhah, , and D. Li, "Characterization and modeling of wireless channels for networked robotic and control systems - a comprehensive overview," in *IEEE International Conference on Intelligent Robots and Systems*, 2009.

[30] C. Rasmussen and C. Williams, *Gaussian Processes for Machine Learning*. The MIT Press, 2006.

[31] J. Candela and C. Rasmussen, "A unifying view of sparse approximate Gaussian process regression," *Journal of Machine Learning Research*, vol. 6, pp. 1939–1959, 2005.

[32] J. Joseph, F. Doshi-Velez, A. S. Huang, and N. Roy, "A bayesian nonparametric approach to modeling motion patterns," *Autonomous Robots*, vol. 31, no. 4, pp. 383–400, 2011.

[33] R. Brafman and M. Tennenholtz, "R-Max - A General Polynomial Time Algorithm for Near-Optimal Reinforcement Learning," *Journal of Machine Learning Research*, vol. 3, pp. 213–231, 2002.

[34] I. Chads, G. Chapron, M.-J. Cros, F. Garcia, and R. Sabbadin, "Markov decision processes (MDP) toolbox." http://www7.inra.fr/mia/T/MDPtoolbox/MDPtoolbox.html, 2012.

[35] J. P. How, E. Frazzoli, and G. Chowdhary, "Linear Flight Contol Techniques for Unmanned Aerial Vehicles," in *Handbook of Unmanned Aerial Vehicles* (K. P. Valavanis and G. J. Vachtsevanos, eds.), Springer, 2014. In Press.

[36] M. Cutler and J. P. How, "Actuator constrained trajectory generation and control for variable-pitch quadrotors," in *AIAA Guidance, Navigation, and Control Conference (GNC)*, (Minneapolis, Minnesota), August 2012.

[37] M. Quigley, B. Gerkey, K. Conley, J. Faust, T. Foote, J. Leibs, E. Berger, R. Wheeler, and A. Ng, "Ros: an open-source robot operating system," in *Proceedings of the Open-Source Software Workshop of the International Conference on Robotics and Automation*, 2009.