Human Aware UAS Path Planning in Urban Environments using Nonstationary MDPs

Rakshit D Allamraju
Mechanical and Aerospace Engineering
Oklahoma State University
rakshit.allamraju@okstate.edu

Hassan A Kingravi
Oklahoma State University
and Georgia Institute of Technology
hkingravi@yahoo.com

Allan M Axelrod
Oklahoma State University

Dr. Girish V Chowdhary
Oklahoma State University
girish.chowdhary@okstate.edu

Robert Grande
Laboratory for Information and Decision Systems
Massachusetts Institute of Technology

Dr. Christopher J Crick
Department Of Computer Science
Stillwater, O.K.

Dr. Weihua Sheng
Electrical and Computer Engineering
Oklahoma State University

Dr. Jonathan How
Laboratory for Information and Decision Systems
Massachusetts Institute of Technology

Abstract

We present a model based reinforcement learning algorithm for solving nonstationary Markov Decision Process that arise in human aware UAS planning. The presented algorithm is flexible enough to accommodate changes in reward function densities, real-time computable, and yet it can improve performance through long term learning using non-Bayesian Gaussian process clustering. The approach is validated experimentally on a large-scale long duration experiment with 5500 runs with both simulated and real UAVs on the problem of avoiding human population densities.

1 Introduction

Most unmanned Aerial Systems (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]). It would therefore be beneficial to create human-aware UAS path planning algorithms that minimize the likelihood of UAS flying over areas with high human density. In order to create a human-aware UAS path, the UAS must be able to plan in anticipation of where humans could be given given the time of the day, the time of the year, and other seasonal considerations. The resulting problem is essentially nonstationary in nature due to the variation in human population densities.

We propose a non-stationary (MDP) formulation of the human-aware path planning problem, and propose a model-based reinforcement learning approach as a solution to the problem. The non-stationarity in our formulation is induced by the time-varying yet periodic nature of human population densities. Our algorithm learns and maintains a separate model for each distinguishable distribution of human density. We use a Gaussian Process 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 of Grande et al. [3] is used to cluster and classify GP cost models in real-time, and a model-based policy iteration algorithm [4] is used to solve the MDP associated with each reward model. Additionally, we introduce a *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 by providing the first GP based BNP non-stationary MDP solution architecture that explicitly handles the trade-off between exploration and exploitation in dynamically evolving environments.

2 Related Work

Gaussian processes have previously been used successfully in model-based reinforcement learning and approximate dynamic programming [5,6]. The main attraction of these approaches has been the flexibility afforded by the GP BNP, 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 [7]. However, existing work on GP-based planning, and other model-based reinforcement learning work [8,9], 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 [10]. Even efficient online versions of existing GP inference algorithms update of the predictive variance independent of the measurements, hence the predictive variance can decrease even when the underlying generative model is changing [11]. Pérez-Cruz et al. [12] and Chowdhary et al. [13] 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 a forgetting factor 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 [3] that detects changes in the underlying generative model and is capable of detecting when a previously-seen model appears again. Additionally, we introduce a *fog of war* concept to ensure sufficient and controllable exploration in an exploration-exploitation framework. These algorithms form the foundation of our nonstationary MDP architecture. Nonstationary MDPs have been previously explored, but the focus has either been on regret bounds [14], or on heuristics to account for the nonstationarity [15]. In summary, our main contributions to the literature include a novel change-point detection algorithm which is capable or learning models online and a novel fog of war exploration strategy. More generally, our architecture provides a widely applicable structured and flexible approach to solve nonstationary MDPs when the underlying reward models and their switching times are unknown by using GP BNP based model learning with a nonstationary detection and variance based exploration strategy.

3 Gaussian Process Regression and GP clustering

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 [16]. A GP is a distribution over functions, that is, a draw from a GP is a function. For the sake of clarity of exposition, we will assume that $\Delta(z) \in \mathbb{R}$; the extension to the multidimensional case is straightforward. When Δ follows a Gaussian process model, then

$$\Delta(\cdot) \sim \mathcal{GP}(m(\cdot), k(\cdot,\cdot)), \tag{1}$$

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

$$\mathcal{G} = \left\{ g(\cdot) \in \mathbb{R}^{\mathcal{X}} \;\middle|\; g(\cdot) = \sum_{i=1}^{\infty} \alpha_i k(z_i, \cdot) \right\}, \tag{2}$$

where $\mathcal{X} = \mathbb{R}^n$, $\alpha_i \in \mathbb{R}$, $z_i \in \mathcal{X}$. The space $\mathcal{G}$ is a subspace of $\mathcal{H}$, an RKHS, and $\|g\|_{\mathcal{H}} < \infty$ where $\|g(\cdot)\|^2_{\mathcal{H}} = \sum_{i=1}^{\infty}\sum_{j=1}^{\infty} \alpha_i \alpha_j k(z_i, z_j)$. This assumption imposes a smoothness prior on the mean and makes the problem amenable to analysis though the representer theorem [17].

Let $Z_\tau = \{z_1, \dots, z_\tau\}$ be a set of state measurements, discretely sampled where $\{1 \dots \tau\}$ are indices for the discrete sample times $\{t_1, \dots, t_\tau\}$. The set defines a covariance matrix $K_{ij} := k(z_i, z_j)$. The positive definite function k generates a mapping ψ to an RKHS $\mathcal{H}$ such that $k(z_i, z_j) = \langle \psi(z_i), \psi(z_j) \rangle_{\mathcal{H}}$. GP regression assumes that the uncertainty in the data and the model follow Gaussian distributions, while modeling the function estimate using a mean function $\hat{m}$ and a covariance function $\hat{\Sigma}$. Since the observations are Gaussian, the likelihood function $p(y_\tau | Z_\tau, \beta)$ is Gaussian. The initial prior is set to $p(\beta) \sim \mathcal{N}(\mathbf{0}, \mathbf{\Sigma_w})$, and Bayes' rule is used to infer the posterior distribution $p(\beta | Z_\tau, y_\tau)$ with each new observation. Since the posterior is Gaussian, the update generates a revised mean $\hat{m}_\tau$ and covariance $\hat{\Sigma}_\tau$. If $|Z_\tau|$ is finite, the solution for the posterior mean and covariance is also finite [17]. In particular, given a new input $z_{\tau+1}$, the joint distribution of the data available up to τ and z_τ under the prior distribution is

$$\begin{bmatrix} y_\tau \\ y_{\tau+1} \end{bmatrix} \sim \mathcal{N}\left(0, \begin{bmatrix} K(Z_\tau, Z_\tau) + \omega^2 I & k_{z_{\tau+1}} \\ k^T_{z_{\tau+1}} & k^*_{\tau+1} \end{bmatrix}\right), \tag{3}$$

where $k_{z_{\tau+1}} = K(z_{\tau+1}, Z_\tau)$ and $k^*_{\tau+1} = k(z_{\tau+1}, z_{\tau+1})$. The posterior (sometimes called the predictive) distribution, obtained by conditioning the joint Gaussian prior distribution over the observation z_{t+1}, is computed by

$$p(y_{\tau+1} | Z_\tau, y_\tau, z_{\tau+1}) \sim \mathcal{N}(\hat{m}_{\tau+1}, \hat{\Sigma}_{\tau+1}), \tag{4}$$

where

$$\hat{m}_{\tau+1} = \beta^T_{\tau+1} k_{z_{\tau+1}} \tag{5}$$

$$\hat{\Sigma}_{\tau+1} = k^*_{\tau+1} - k^T_{z_{\tau+1}} C_\tau k_{z_{\tau+1}} \tag{6}$$

are the updated mean and covariance estimates, respectively, and where $C_\tau := (K(Z_\tau, Z_\tau) + \omega^2 I)^{-1}$ and $\beta_{\tau+1} := C_\tau y_\tau$.

Since both Z_τ and y_τ grow with data, computing the inverse becomes computationally intractable over time. Therefore, to adapt GPs for the online setting, we use the efficient and recursive scheme introduced in [11]. The set Z generates a family of functions $\mathcal{F}_Z \subset \mathcal{H}$ whose richness characterizes the quality of the posterior inference; therefore 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. This is known as the kernel linear independence test [11], and is computed by

$$\gamma_{\tau+1} = \left\| \sum_{i=1}^{\tau} \alpha_i \psi(z_i) - \psi(z_{\tau+1}) \right\|^2_{\mathcal{H}}. \tag{7}$$

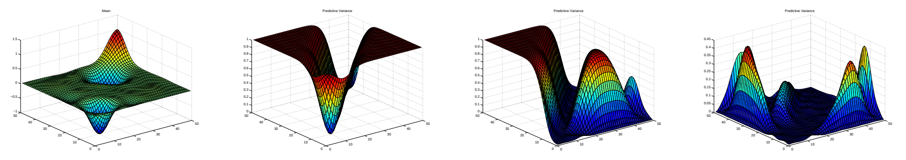
The scalar $\gamma_{\tau+1}$ is the length of the residual of $\psi(z_{\tau+1})$ projected onto the subspace $\mathcal{F}_{Z_\tau}$. When $\gamma_{\tau+1}$ is larger than a specified threshold, then a new data point should be added to the data set. The coefficient vector α minimizing (7) is given by $\alpha_\tau = K^{-1}_{Z_\tau} k_{z_{\tau+1}}$, meaning that

$$\gamma_{\tau+1} = k^*_{\tau+1} - k^T_{z_{\tau+1}} \alpha_\tau. \tag{8}$$

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.

3.1 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. Grande et al. [3] presented such an algorithm in context of regression and clustering. Here we use it for solving nonstationary MDPs. The details are avoided here due to space constraints.



(a) Predictive mean (b) Predictive variance at t_1 (c) Predictive variance at t_5 (d) Predictive variance at t_{20}

Figure 1: An example of how the predictive variance decreases in GP inference; as can be seen, as more and more of the space is explored, the predictive variance decreases globally.

4 Proposed Solution to Nonstationary MDPs

In this section, we leverage the above ideas to create an algorithm for nonstationary MDPs with unknown reward models. In the algorithm, the GP clustering algorithm of [3] 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.

4.1 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. Figure 1 shows an example of this concept in action. In the example, a GP is initialized over a domain $D \subset \mathbb{R}^2$, and an agent explores the space, using samples to update the GP. As can be seen, at t_1, the predictive variance is extremely high over the space; by time t_{20} however, it has reduced greatly because the agent has explored most of the space. 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

$$\hat{r}_{e_{t_i}}(x) = \hat{\Sigma}_{t_i}(x) + \Upsilon_{t_i}(x), \tag{9}$$

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 (9). 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

$$s_e(t_i) = \kappa - \frac{1}{\mathrm{vol}(D)} \int_D \hat{r}_{e_{t_i}}, \tag{10}$$

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 [18] and information entropy maximizing active sensor placement algorithm [7]. 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 in (6) 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 5.

Putting it all together, we get the algorithm shown in Algorithm 1.

Algorithm 1 Nonstationary MDP Solver

```
Initialize: Initial data $(X, Y)$, lps size $l$, model deviation $\eta$, GP parameters $(\sigma, \omega_n^2)$, space exploration threshold $\varphi$.
while new data $(x_i, y_i)$ is available do
  Update GP cluster using [3].
  Compute exploration reward $\hat{r}_{e_{t_i}}$ using (9).
  Compute space explored $s_e$ using (10).
  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
```

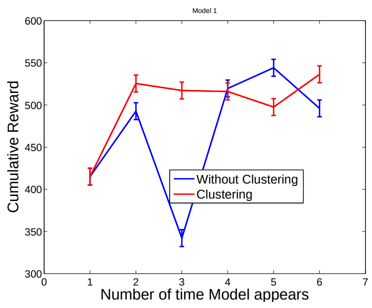
5 Experimental Results

5.1 Description of the Experiment

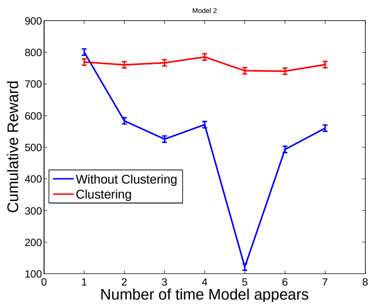
The Goal of the experiment is to validate the performance of the proposed planner. The mission scenario is for the agents to go from a predefined ingress location to a predefined egress location (referred to as the goal location) on the domain. The agent is free to choose the optimal path over the domain to perform its mission. Each path planning and execution instance is termed as a run. A large-scale experiment with a total of 5500 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 [19] 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 (5500), both real and 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 [20,21].

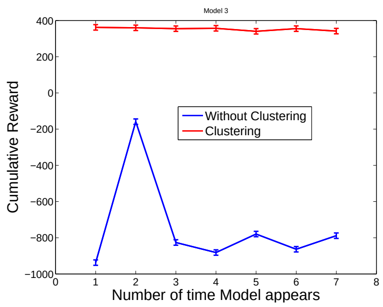
Figure 2 presents the average reward accumulated and its variance by both the planners for each of the four models. The horizontal axis indicates the number of times each model is active and the vertical axis indicates the cumulative reward the agent has accumulated over the time the underlying model was active. In model 1, both algorithms perform very similarly. One reason for this is because GPR tends to learn model 1 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



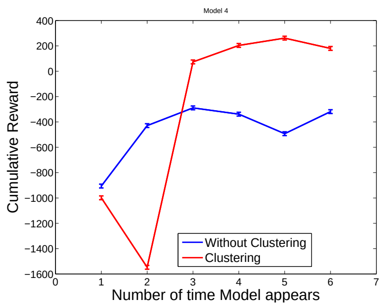
(a) Model 1



(b) Model 2

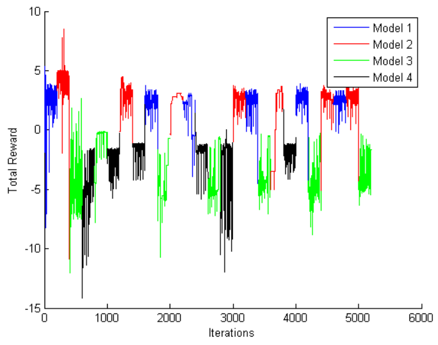


(c) Model 3

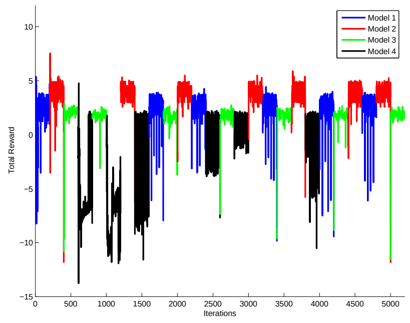


(d) Model 4

Figure 2: Comparison between Total Rewards for each model



(a) Cumulative Reward without Clustering



(b) Cumulative Reward with Clustering

Figure 3: Comparison of the accumulated rewards of GP clustering against GPRegression. Agent Accumulates positive rewards when clustering models as compared to GPRegression.

the small variations being attributed to the stochasticity in the reward. Figure 3 indicates total reward accumulated by the agent in each iteration, when the underlying reward model is changing. Figure 4 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

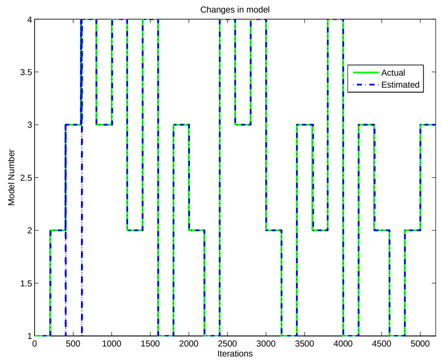
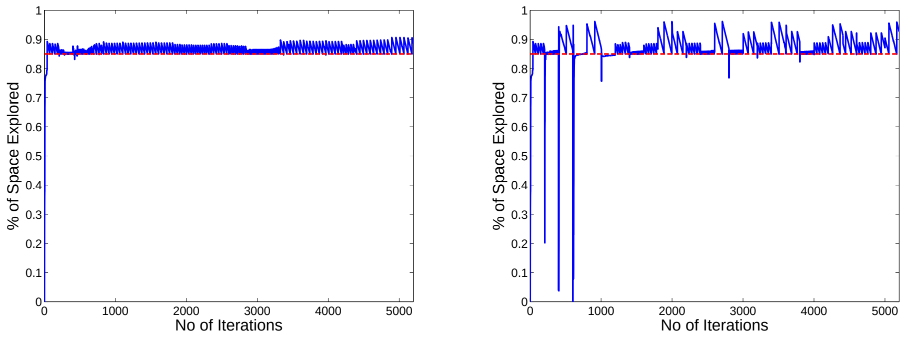


Figure 4: Plot indicating the actual model being tracked by the estimated model. At the 200^{th} and 400^{th} run new models are introduced, the algorithm quickly detects them after a brief misclassification. Note that the algorithm clusters the underlying reward model quickly and reliably afterwards.



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

Figure 5: Space Explored by each planner indicative of the variance

algorithm to ensure that it obtains sufficient information to compute the optimal policy [22]. In our architecture, the exploration strategy is to compute a policy that guides the agents to areas of the domain where it has little confidence in its model. However, as discussed in Section 4.1, relying simply on the GP variance is not enough, because the GP variance updates in closed-form GP algorithms ([10,11]) do not take into account the non-stationarity in the domain. The fog of war forgetting introduced in (9) 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 to explore as much. It is interesting further to note that the learned reward models are most accurate along the optimal trajectory, and only approximately accurate on paths that are not optimal. This is another indication of optimality, in the sense that the algorithm does not spend time exploring areas where it cannot perform optimally. Furthermore, Figure 5 plots the value of the exploration reward (10) 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.

6 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] R. Grande, T. Walsh, S. Fergusson, G. Chowdhary, and J. How, "Online regression for non-stationary data using gaussian processes and reusable models," in *International Conference on Machine Learning*, 2013. submitted.

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

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

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

[7] 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.

[8] 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.

[9] 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.

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

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

[12] 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.

[13] 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).

[14] 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.

[15] 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.

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

[17] B. Scholkopf, R. Herbrich, and A. Smola, "A generalized representer theorem," in *Computational Learning Theory* (D. Helmbold and B. Williamson, eds.), vol. 2111 of *Lecture Notes in Computer Science*, pp. 416–426, Springer Berlin / Heidelberg, 2001.

[18] 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.

[19] 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.

[20] 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.

[21] 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.

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