Decentralized and Learned Search Parameters For Cooperating Vehicles using Road Network Gaussian Process Regressions

Brady G. Moon and Cameron K. Peterson

***Abstract*— Unmanned aerial vehicles are capable of working as teams to accomplish a wide variety of mission objectives, such as searching for and tracking targets. In this paper, vehicle search paths were dictated by a joint cost function which maximizes the reward earned by searching a road network. In previous work, these rewards were assigned based on the elapsed time since the section had last been searched. This approach is effective in rewarding vehicles to search out areas which haven't been visited in a long time, but it lacks the ability to weight roads differently based on the probability that targets will be in that section. This paper proposes a method of using accumulated knowledge of the average density of targets along a road section, along with a Gaussian process regression to assign search rewards. The target density information that drives the Gaussian process regression is propagated through the vehicle communication network using decentralized consensus filters. Vehicles then choose paths that are more likely to find targets rather than seeking areas which have not been searched recently. Through numerical simulations we show that this method increases the number of targets seen by cooperating UAVs and provides an accurate estimate of target density along a road network. CKP: Quantify how much improvment?**

I. Introduction

The high demand for unmanned aerial vehicles (UAVs) continues to rise due to their many desirable applications and qualities. They have a multitude of practical uses such as patrolling [1], target tracking [2], search and rescue [3], mapping dangerous regions [4], [5], and surveillance [6]. Additionally, UAVs are usually cheaper than conventional methods, such as manned aircraft, and keep the pilots out of harm's way [7].

Patrolling, searching, and surveillance objectives all face challenges when the available UAVs are insufficient for viewing the entire area simultaneously. In these cases, the UAVs must coordinate their trajectories to optimally cover the largest area possible. This task becomes increasingly difficult when either the search area becomes large or there is a limitation in the number of available UAVs.

The effectiveness of a search method is evaluated by the overall mission goal, such as equal and frequent coverage of an area or finding and monitoring moving targets within the region. Each overall goal has its benefits and drawbacks. Frequency-based algorithms focus on optimizing the amount of elapsed time between visits to locations in the search area. Some seek to minimize this elapsed time, as in [8], [9], while others work towards making the elapsed time uniform across the search grid, as seen in [10]. Another approach is to maximize the number of high-interest points found in the least amount of time. This utilizes a more probabilistic algorithm, as seen in [11]. Reinforcement learning has been introduced as an aspect to minimize the search area coverage rate and has been been examined in [12], [13].

Although these methods are effective in their evaluation criteria, they have weaknesses in the application of searching for and monitoring moving targets. They fall short in maximizing the number of targets discovered and tracked. In many situations, targets are not uniformly distributed across a search area and UAVs spend time searching areas where there are likely to be few targets. This paper presents a method of using learned knowledge from the environment to maximize the number of detected targets. In this approach, UAVs have higher revisit rates in areas where they are more likely to gain information on new or existing targets, and they decrease time spent in areas where targets are rarely observed.

This result is achieved by balancing the exploration and exploitation of learned information about target densities and patterns through the use of Gaussian Process (GP) regressions. GP regressions are an effective way to predict Gaussian processes [14]. GP regressions have been used in previous research for estimating wind fields for gliders and UAVs [15], [16], pattern discovery [17], and predictive controls in UAVs [18], [19].

An important aspect of multi-agent path planning is to provide algorithms capable of scaling to the increased number of vehicles. This generally necessitates decentralized algorithms that avoid communicating large amounts of information to peer agents or a centralized ground station. In this paper, an estimate of the learned target densities will be decentrally derived using consensus filters. Consensus filters are useful in driving vehicles to a common state [20]. In prior research, they have been used for estimation of environmental information [21], [22], tracking objects [**?**], determining the number of targets [23], and driving vehicles to desired formations [cite]. In this work, it is assumed that the UAVs have a finite communication radius and are unable to connect back to a centralized processing station. Each UAV shares information about the number of vehicles it has detected and the number of times it has viewed a road segment. We propagate this information from each UAV to all peer UAVs through a decentralized method based upon consensus filters. We assume that each UAV communication connection is undirected and that the entire set of UAVs are connected.

The overarching objective of this paper is to answer the question of how we can improve the ability of cooperat-

Brady G. Moon is an undergraduate student in Electrical Engineering at Brigham Young University, bradygmoon@gmail.com

Cameron K. Peterson is with the Department of Electrical Engineering, Brigham Young University Provo, UT USA, cammy.peterson@byu.edu

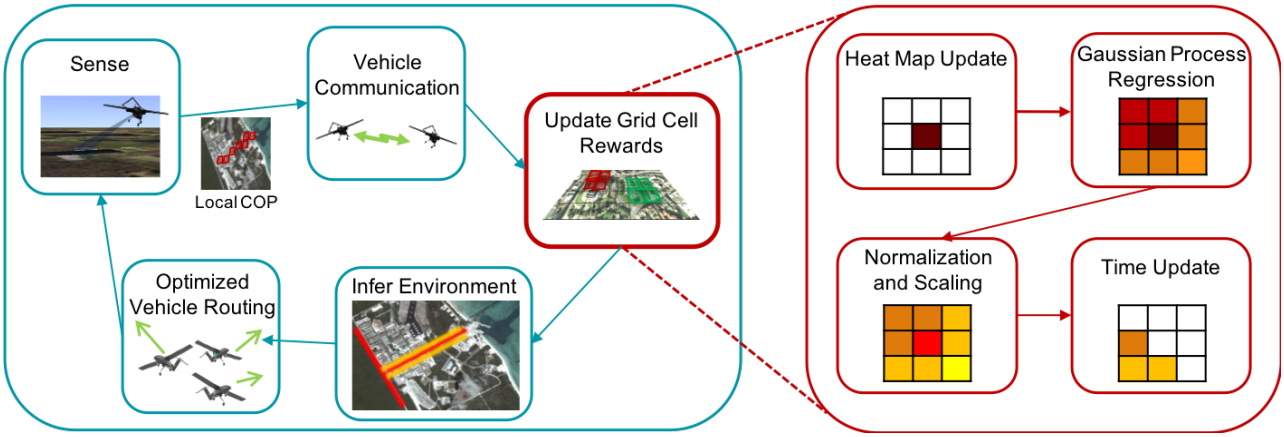


Fig. 1. The functional diagram of the cooperative search discrete simulation. This shows that once vehicles sense information from their environment and share their local common operating picture then this information is incorporated in the grid cell rewards.

ing UAVs to detect moving targets when operating in an unknown region. The approach for answering this objective include both the development of a new search method and a process of incorporating infrastructure information, in the form of open-source road-networks, for use in this search algorithm. Searching is achieved in a bounded region through a combined use of GP regressions, which drive the information gathering, and target density heat maps, which characterized the best understanding of the density within the environment. The heat map integrates its calculated average target density with the UAV's path planning cost function by using it to dynamically modify the reward value given for searching. The modified reward value is used to cooperatively decide each UAV's desired path.

This paper is an extension of the authors prior published conference proceeding [24] which applied GP regressions in a centralized implementation to a bounded search region by segmenting the area into grid cells. The road network approach presented in this publication provides a number of advantages over a grid cell approach, including (a) the ability to automatically learn the road network through open source information, (b) the ability to limit UAVs from searching areas where targets will never be found (i.e. off of roads), (c) a more accurate prediction of target locations, and (d) the ability to propagate target uncertainty along the road network through a novel kernel function applied in the GP regression.

Specifically, the contributions of this paper include:

1) The development of a search algorithm that increases the detection of moving targets constrained to a road network.
2) The automatic creation of a road network using open-source databases.
3) The use of a novel kernel function that determines spatial closeness to points along the network and thus provides a method to update the GP regression.
4) The use of consensus filters to construct a decentralized implementation of the search algorithm.

This method thrives in search areas where target location densities and frequencies are nonuniform. For example, a search area which has a mix of downtown and suburban areas, or sparsely distributed roads. As the UAVs learn where they are more likely to see targets, they adjust their paths to visit these areas more often. This method was tested and compared against two baseline approaches, that will be described in Section IV, to validate its utility.

The paper proceeds as follows. Section II describes the overall path planning approach and how cooperating vehicles are driven to optimize their search area using a cost function. The main contributions of this paper are outlined in Section III, where the method of calculating and modifying the grid cell rewards from learned information is described. The comparisons of this method with baseline approaches are given in Section IV. And Section V provides conclusions and a summary.

CKP: Need to be a bit clearer about the type of data we are using in the simulations in the intro.

II. Vehicle Path Planning

This section describes the path planning algorithm and how it enables cooperative control among UAVs. Fig. 1 shows the main components of the simulation framework, where each vehicle is capable of gathering information from within its sensing field of view. Using that information it forms a local common operating picture (COP) that encapsulates its learned knowledge of the environment. The COP information is communicated with peer vehicles and used to update the parameters of a search reward-function. The vehicle's best understanding of the environment is propagated forward and then used to cooperatively compute the UAVs paths. Once all the vehicles move forward in time this discrete process is repeated.

In the remainder of this section, the UAV model will be explained (Section II-A), as well as the method for choosing vehicle paths using a receding horizon control (Section II-B).

A. Vehicle and Target Model

All UAVs are modeled as fixed wing aircrafts that are controlled by commanding their roll angles. It is assumed

that each UAV travels at a constant velocity and altitude and that changes in heading follow a coordinated turn model

$$\dot{\psi}_v(t) = \frac{g \cdot \tan \phi_v(t)}{V_v(t)},$$

where $\phi_v(t)$ is the roll angle, g is the gravitational constant, $V_v(t)$ is the UAV speed at time t, and $\dot{\psi}_v(t)$ is the turn rate. All aircrafts have a saturation limit on the amount they can roll by $|\dot{\phi}_v| \leq \dot{\phi}_{max}$.

The target state of the pth vehicle is given by $\boldsymbol{x}_p = [x_p \ y_p \ \dot{x}_p \ \dot{y}_p]^T$ and contains the target's Cartesian position and velocity. Aircraft are equipped with a sensor capable of detecting ground targets within their sensing radii. UAV sensors measure range and azimuth, therefore detections must be converted from this measurement space to the Cartesian state space. The measurement noise $\boldsymbol{w}_v$ of vehicle v is parameterized by range and azimuth uncertainty σ_r and σ_θ and measurement noise covariance $R = diag[\sigma_r^2, \sigma_\theta^2]$. The measurement matrix, which relates measurement space to state space, is given by the the Jacobian

$$H_{v,p}(k) = \begin{bmatrix} \frac{x_p(k)-x_v(k)}{r_{p,v}(k)} & \frac{y_p(k)-y_v(k)}{r_{p,v}(k)} & 0 & 0 \\ \frac{y_p(k)-y_v(k)}{r_{p,v}(k)^2} & -\frac{x_p(k)-x_v(k)}{r_{p,v}(k)^2} & 0 & 0 \end{bmatrix},$$

where the planar position of each vehicle is represented at time step k as $(x_v(k), y_v(k))$, each target is positioned at $(x_p(k), y_p(k))$, and $r_{p,v}(k) = \sqrt{(x_p(k) - x_v(k))^2 + (y_p(k) - y_v(k))^2}$ is the Euclidean distance between them.

Cartesian measurements are modeled as $z_{v,p}(k) = H_{v,p}(k)\boldsymbol{x}_p(k) + \boldsymbol{w}_v$ and estimates of the target's state are updated using a Kalman filter as described in [25]. Detections on targets are shared with all other cooperating vehicles.

B. Search Mission and Vehicle Routing

BM: Is this the section for probabilistic target road association? Also, naming of variables has now been modified in this paragraph. I need to propagate these changes throughout the rest of the paper.

A search mission is implemented by rewarding vehicles that search road segments. The UAVs are directed to search an area $M \in \mathbb{R}^2$. The road map within M is obtained through exporting the road data of the area of interest from the OpenStreetMap database. Major roads are extracted from the data, while discarding trails, paths, and other lower-interest road segments to create a set of road segments S. In order for all the road segments to be relatively the same length, a maximum and minimum road segment length is chosen, with the maximum road segment length needing to be less the the sensing diameter of the UAVs. The selected road segments which are greater than the specified maximum road segment length are divided into equal sections such that each segment is then under the maximum road segment length. Newly created segments are appended to the set S. Road segments in S which lengths are less than the specified minimum road segment length are grouped with adjacent road segments to create G, where G is made up of subsets of S where all segments of S are contained in G and any G_i is a subset of S where its segments are not contained in any other G_i and the total length of road segments within G_i is greater than the minimum road segment length.

The reward function for a UAV is

$$J_{search} = \sum_g J_g(k), \quad \forall g \in \Gamma,$$

where Γ is the union of groups that have all of their segment endpoints contained within the vehicles sensing radius BM: Update the following equation, $r_g = \sqrt{(x_g - x_v)^2 + (y_g - y_v)^2} < r_s$ and $J_g(k)$ is the reward of the gth group. $J_g(k)$ is time-varying and may be different for each group. The process of defining $J_g(k)$ is the topic of Section III.

A receding horizon control (RHC) is used to jointly plan the paths of all the vehicles. RHCs work by choosing a path that maximizes the reward function over an event horizon. The vehicles then move forward one step along their decided paths and the process iterates with a reevaluation of the reward function to the next event horizon. At each time step in its horizon, the vehicle evaluates all its potential control commands. For this paper, each UAV commands its roll angle, ϕ_c, and may choose to bank left, go straight, or bank right, i.e. $\phi_c \in [-\phi_{max}, 0, \phi_{max}]$.

A joint-search RHC must maximize the reward given for the combined UAV paths. The group reward is computed using [26]

$$J^V_{search} = \sum_g J_g(k), \quad \forall g \in \Gamma^V,$$

where $\Gamma^V = \{\bigcup_v^V (r_g < r_s)\}$ is the set of grid points that lie within every vehicle's sensing radius. The grid cell value, $J_g(k)$, is rewarded if it is contained within the field of of view of any vehicle. Thus, viewing the same grid point simultaneously by multiple vehicles yields no additional reward.

Joint path planning is a computationally expensive process. When completed using an exhaustive search, the reward for every combination of potential UAV paths must be evaluated. This approach becomes intractable even with moderate numbers of vehicles and short event horizons. To alleviate computational cost a Rollout policy is used to reduce the number of paths evaluated.

The Rollout policy uses a combination of exhaustive search with a greedy-heuristic algorithm. Initially each possible combination of command decisions are evaluated. However, after a specified number of time steps only the immediate best reward is greedily chosen. This drastically reduces the number of potential UAV paths (and therefore combinations of paths) that must be evaluated.

This paper uses a variation of the Rollout policy presented in [6]. It is augmented with an adaptable threshold that determines when to switch from the exhaustive search to the greedy-heuristic algorithm. The criteria is based off the separation between concurrent UAV paths and forces spatial diversity between the command decisions being evaluated.

III. Determining Road Segment Rewards

This section describes the method of setting the road segment values which are awarded to a UAV if it searches that road segment. The high level objective for all UAVs is to search the operational region in a manner that maximizes track detections on the largest number of targets. Therefore the individual road segment values must incentivize vehicles to search road segments that contain high target densities. To accomplish this goal, reward values are dynamically modified to fluctuate with the predicted target densities on each segment.

The process for computing road segment rewards is seen in the right half portion of Fig. (1). The steps, which will be expounded upon in the subsequent subsections, are as follows:

1) Local Heat Map Update: Using a probabilistic approach, sensed target are attributed to road segment groups within the sensing radius. BM: Update and move. We modified code slightly. consensus across which things and when. Subtract to find current update or something.
2) Consensus: Target counts and the number of times a segment has been seen are shared using a consensus algorithm. Road segment densities and variances are updated according to each vehicles' individual knowledge.
3) Gaussian Process Regression: Each vehicle performs a GP regression on the road segments using their density and variance calculations.
4) Normalization and Scaling: The GP regression estimates are increased by their standard deviations and then normalized to a predetermined maximum J_{max}. The normalized counts are assigned to each grid cell as their upper reward limit, $J_{g,max}$ BM: Update notation.
5) Time Update: The current reward values for all cells are calculated using an exponential function which increases beginning from the time the road segment was last searched.

A. Local Heat Map Update

BM: Do explain how information spread within group. Also make sure explanation of density vs amount is correct. This wasn't emphasized last paper because grid cells used to be the same size so density vs predicted about within didn't matter. Even target density accros road segmnets within group. We are not predicting density but counts.

At each time step, all vehicles updates their local heat map with the current target tracks. In order to coordinate targets with the road they are currently driving on, a probablisitc approach is taken. BM: CKP insert Probabilistic target road association stuff here.

After the blah is spread, then Only assign targets to roads that group fully seen. Local array. Add to total counts and total times seen. These are the two consesnus is across. Array H where H_s is blah at segment s. Both needed? Yes, we do both. maybe not necessary. BM: Check calculation for finding how many times a segment has been seen

Information of each road segment is saved in a one-dimensional array C_s and variable T_s. Each entry of C_s contains the number of targets assigned to each road segment through the probabilistic target road association when its road segment group was sensed by a vehicle. T_s is the number of times that road segment group has been viewed. Whenever a road segment group g falls within a vehicle's sensing radius,.....add up total wihtin the group, and distributed proportional to segent length withinthe group. How say that better. Then does the part following. .... for each road segment s within that group, the number of targets detected and associated with that segment are appended to C_s, and T_s is incremented. At the beginning of each simulation, C_s and T_s are populated with non-zero target counts to encourage an initial exploration of all road segments. The average targets in each cell over time is calculated as

B. Consensus

Consensus algorithm and method here.

Deduce how many were seen at that point and update the arrays explained below.

Information of each road segment is saved in a one-dimensional array C_s and variable T_s. Each entry of C_s contains the number of targets assigned to each road segment through the probabilistic target road association when its road segment group was sensed by a vehicle. T_s is the number of times that road segment group has been viewed. Whenever a road segment group g falls within a vehicle's sensing radius,.....add up total wihtin the group, and distributed proportional to segent length withinthe group. How say that better. Then does the part following. .... for each road segment s within that group, the number of targets detected and associated with that segment are appended to C_s, and T_s is incremented. At the beginning of each simulation, C_s and T_s are populated with non-zero target counts to encourage an initial exploration of all road segments. The average targets in each cell over time is calculated as BM: Check this carefully against the Matlab 08/27/19

$$\overline{X}_g = \frac{\sum_{T_g} C_g(k)}{T_g}, \tag{1}$$

and the standard deviation is

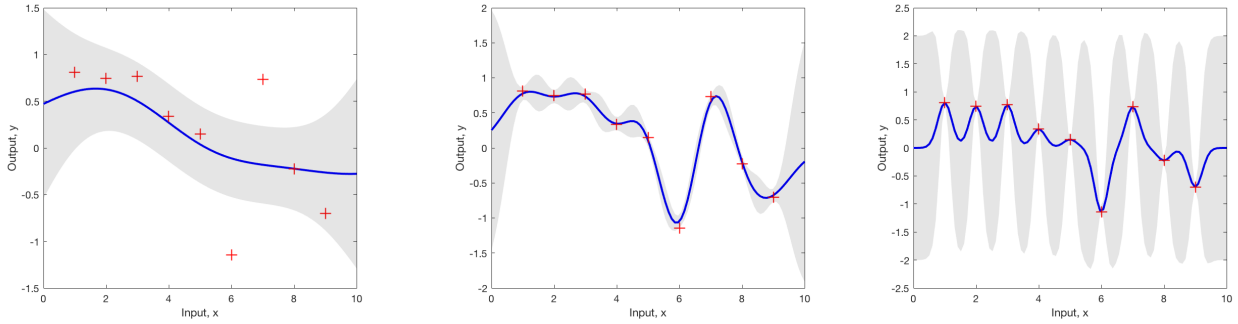
$$s_g = \sqrt{\frac{\sum_{T_g} \left(C_g(k) - \overline{X}_g\right)^2}{T_g}}. \tag{2}$$

The values of $\overline{X}$ create a target density heat map, which reflects its best knowledge of the area. This heat map will continue to change as the UAVs search the area and take more measurements of targets within the domain.

C. Gaussian Process Regression

In this section the aspects of GP regression relevant to this paper are introduced. A more comprehensive treatment of this topic is given in [14].

Using information from the target density heat map, a GP regression is performed across all the grid cells using their updated means and variances, $\overline{X}_g$ and s_g^2. GP is a collection of random variables at input points X that are characterized



(a) Under-fit GP regression (b) Good-fit GP regression (c) Over-fit GP Regression

Fig. 2. Shows an example of how the length factor and σ_f^2 affect the GP regression fit using test data of a sine wave with added noise.

by a mean function $m(X)$ and covariance function $k(X, X')$. The distribution over an arbitrary function $f(X)$ is defined as

$$f(X) \sim GP\big(m(X), k(X, X')\big).$$

A GP regression uses Bayes' rule to compute a posterior distribution over functions based on training and test points. For this work, the training data comes from the grid cell measurements, and the test points are all grid cells within the search area. The posterior distribution is used to infer knowledge about the target density throughout the search region.

It is assumed that $m(X) \equiv \mathbf{0}$, and our input X is defined as all grid cells in S. Our prior covariance matrix K is $n \times n$, where n is the number of grid cells which have at least one measurement. K is constructed using a Laplacian kernel which computes each i, jth component with the function

$$k(S_i, S_j) = \sigma_f^2 e^{-\lambda\left|\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}\right|}, \tag{3}$$

where σ_f^2 is the variance of the signal, λ is the length scale, and $i, j \in \xi$, where $\xi \subset S$ and only contains the grid points which have been measured at least once.

The length scale affects the distance traveled in the input space before the output changes significantly [14]. In a similar manner to [14], Fig. (2) shows how the length factor and signal variance affect the GP regression, where varying their values can create a good-fit, under-fit, or over-fit result. Data points drawn from a noisy sine wave are shown as red + symbols. The blue line is the GP prediction and the gray area represents the 95% confidence region. Panel 2(a) is a GP regression with high σ_f^2 and a length factor which makes it under fit. Panel 2(b) is an example of a GP regression with a lower σ_f^2 and a chosen length factor that provides a good fit. Panel 2(c) shows an over-fit GP regression. The GP prediction follows the training data too perfectly and the region of uncertainty expands out extremely quickly, even in small distances from a training data point.

The hyperparameters, λ and σ_f^2, must be chosen carefully and are optimized once to be kept constant for all simulations. The length scale is adjusted to create a good fit

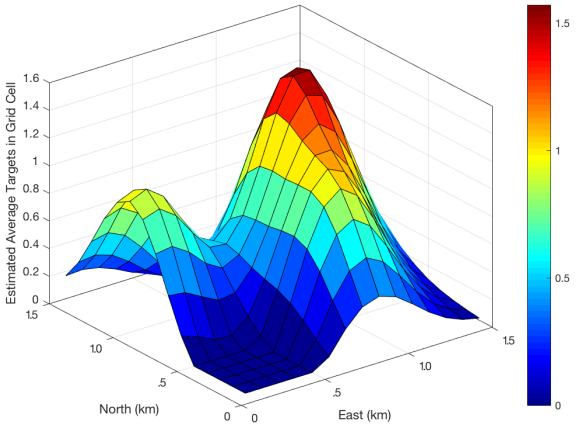


Fig. 3. GP regression of grid cell values.

based on the distances between grid points. The Laplacian kernel has the benefit of being less sensitive to changes in its hyperparameters in comparison to other kernels.

To predict the mean target density and its variance, a joint covariance matrix of current data and desired estimates, K_s, must also be constructed. The matrix K_s is $l \times n$, where l is the total number of grid cells, and n is still the number of grid cells with one or more measurements. It is constructed using equation (3), where $K_s(i, j) = k(S_i, S_j)$, $\forall i \in S$ and $\forall j \in \xi$.

New grid cell predictions are calculated through

$$f = K_s(K + E)^{-1}\overline{X}, \tag{4}$$

where E is a diagonal matrix with elements $E(i, i) = s_i^2$, $\forall i \in \xi$, and s is derived from equation (2) [27], [28]. $\overline{X}$ are the cell averages from equation (1) and f becomes an array of length l. Graphing the GP predictions f produces a result like Fig. (3). This figure is for a 1.5 km square search grid and has sample data at six locations.

The variance of f is calculated by

$$\sigma^2 = K_{ss} - K_s(K + s^2 I)^{-1} K_s^\top, \tag{5}$$

where $K_{ss}(i,j) = k(S_i, S_j)$, $\forall i, j \in S$.

D. Normalization and Scaling

The outputs from the GP regression are the predicted target densities within each grid cell and its uncertainty. To incentivize exploration, we add one standard deviation to the predicted target density according to $Y_g = f_g + \sigma_g$, where $Y = [Y_1, Y_2, \ldots, Y_n]^\top$ and $n \in S$. Initializing C_g and T_g with non-zero values ensures Y_g is not initially zero and the UAVs will have an incentive to explore a grid cell multiple time before the variance is decreased.

The array Y needs to be normalized to maintain a balanced reward function when integrating with other mission objectives, such as target tracking or collision avoidance. Any negative values are set to zero, and each grid cell maximum reward is computed by

$$J_{g,max} = \frac{Y_g J_{max}}{\max(Y)} \tag{6}$$

where J_{max} is the set maximum reward for any grid.

E. Time Update

The reward for viewing a region grows exponentially beginning from the time it was last searched. For a single grid point, g, at time k the reward is [26]

$$J_g(k) = J_{g,max} - (J_{g,max} - J_g(k-1))\, e^{-\Delta t/\Lambda} \tag{7}$$

where Λ is a growth rate, k is the time step, and $J_{g,max}$ is an upper reward limit that the grid cell cannot exceed.

F. Grid Cell Reward Summary

Algorithm 1 shows an overview of the grid cell reward method. This process is repeated at every time step and incorporates the most recently gained knowledge of target positions. Reward values therefore represent the best value for the information gained up to that current time. The UAVs use the reward values to infer their environment and optimize routes, as seen in Fig. (1). Equation (7) is used to compute grid cell values, $J_g(k)$, for future time steps that are used in the RHC.

IV. Results

In this section we use two simulated environments to test the GP regression methodology for calculating grid cell rewards. The first scenario contains both urban and rural areas and illustrates the algorithm's ability to map and explore a mixed environment. The second scenario contains two separate hubs of high density targets with a sparse area in between. It shows that the vehicles do not get trapped in a local maxima by focusing only on one of the high density regions.

Using these simulations, the GP regression method is tested against two alternative methods of searching. The results show that GP regression improves the ability to accurately model the target environment and therefore increases the UAVs tracking capabilities.

The first comparison method uses uniform maximum search rewards (UMSR). UMSR uses equations (6) and (7) with a fixed J_{max} rather than one that changes dynamically based off learned information. Grid cell rewards are then only dependent upon the time since their last update.

The second comparison method lets the learned heat map dictate each cell's maximum grid cell rewards, $J_{g,max}$. This method is called "Heat Map Maximum Search Reward" (HMMSR). It uses the mean target value for each cell to compute the normalized reward. This method lacks the incentive to search regions which were initially found to have a low mean target value. It also lacks the ability to predict mean target values in cells which have not yet been searched.

An overall heat map is created for each search area and updated using the true target positions at every time step. This is calculated by keeping a running sum of targets within each cell and dividing by the current time. Because this heat map uses all target truths, it provides the true heat map for each time step. This allows for numerical comparison between it and the computed heat maps from the three different search algorithms.

These three methods are compared using two different metrics. The first is the fraction of targets seen. This is the number of targets being tracked by the UAVs divided by the total number of targets in the search area. This metric should be high, reflecting that the search method is effective in leading the UAVs to areas with a higher target density. However, the search needs to balance exploitation of dense target areas with exploration of the entire search region. This ensures that the UAVs are following the true densities of the environment, and not getting fixated on a local maximum. Therefore, high accuracy of the UAVs learned heat map of targets in the search area is also desirable.

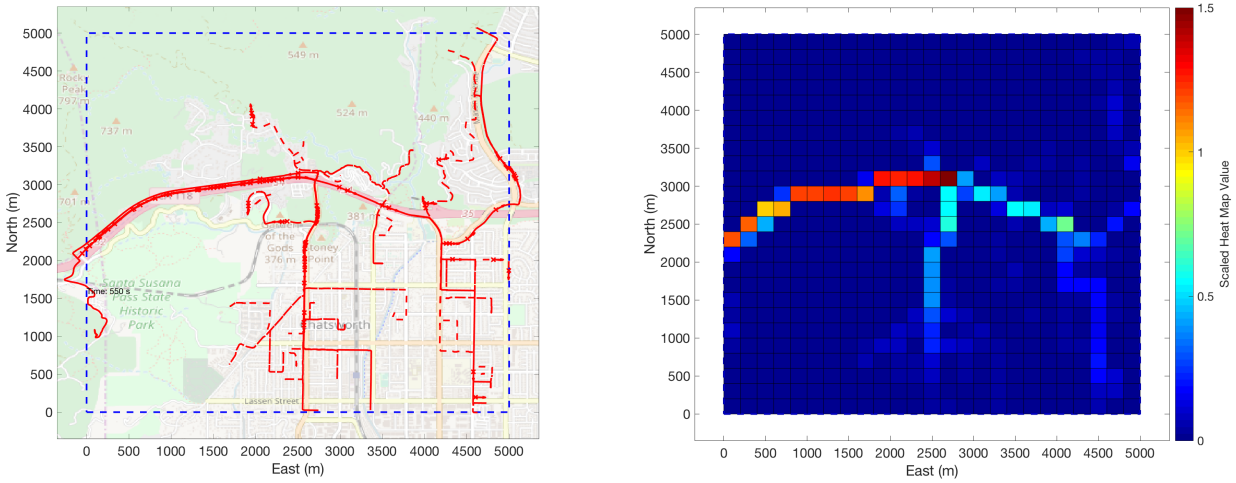
The second metric is the root mean squared (RMS) error of the learned heat maps compared with the overall (true) heat map at each time step. The initial values given to all grid cells to artificially add variability is removed in order to compare only the information learned by the UAVs. New cell target averages are calculated from equation (1). The RMS error then represents the difference in the knowledge of target densities. A lower value reflects a better knowledge of the environment and target densities, which can be used in the future to increase the fraction of targets seen.

A. Chatsworth Search Simulation

Our first simulation environment is seen in Fig. (4). In Panel 4(a), the blue dotted line shows the 5000 meter square search area over a section of Chatsworth, California. The red lines show all of the target paths over the length of the whole simulation. The target trajectories were generated using the Simulation of Urban Mobility (SUMO) software package, which moves each target according to realistic driving patterns and actual road networks [29]. The target densities reflect that more targets are present on larger roads and disperse out to the rural areas. A freeway runs horizontally starting from the middle left side of the search area. There are also two exits from the freeway in the middle and right section of the freeway. An overall heat map of Chatsworth at time $t = 550$ seconds is seen in Fig. 4(b).

Algorithm 1 Method for calculating the grid cell rewards at each time step.

Procedure: GRID CELL REWARD UPDATE
Inputs: w = grid cell width, h = grid cell height, $x1$ = western search boundary, $y1$ = southern search boundary
Outputs: grid cell reward J_g at time k
Make array $targetsInRange$ of all target (x, y) locations within sensing radii of UAVs
Compute i, the horizontal cell number of the location of each target using $targetsInRange$
Compute j, the vertical cell number of the location of each target using $targetsInRange$
Use i and j to update *counts* within each grid cell
For all grid cells g within sensing radii of UAVs
Increment T_g
Append $counts(g)$ to the array of previous counts C_g
Compute average targets in cell, $\overline{X}_g$ (Equation (1))
Compute target-count variance, s_g^2 (Equation (2))
end for
GP Regression:
Compute covariance matrices K, K_s, K_{ss} (Equation (3))
Find estimations f and variances σ^2 (Equations (4) and (5))
$Y_g \leftarrow f_g + \sigma_g$
Normalization and Scaling:
Normalize Y_g (Equation (6)) to find $J_{g,max}$
Time Update:
Find grid cell reward J_g at time k (Equation (7))



(a) Chatsworth search simulation with target paths.

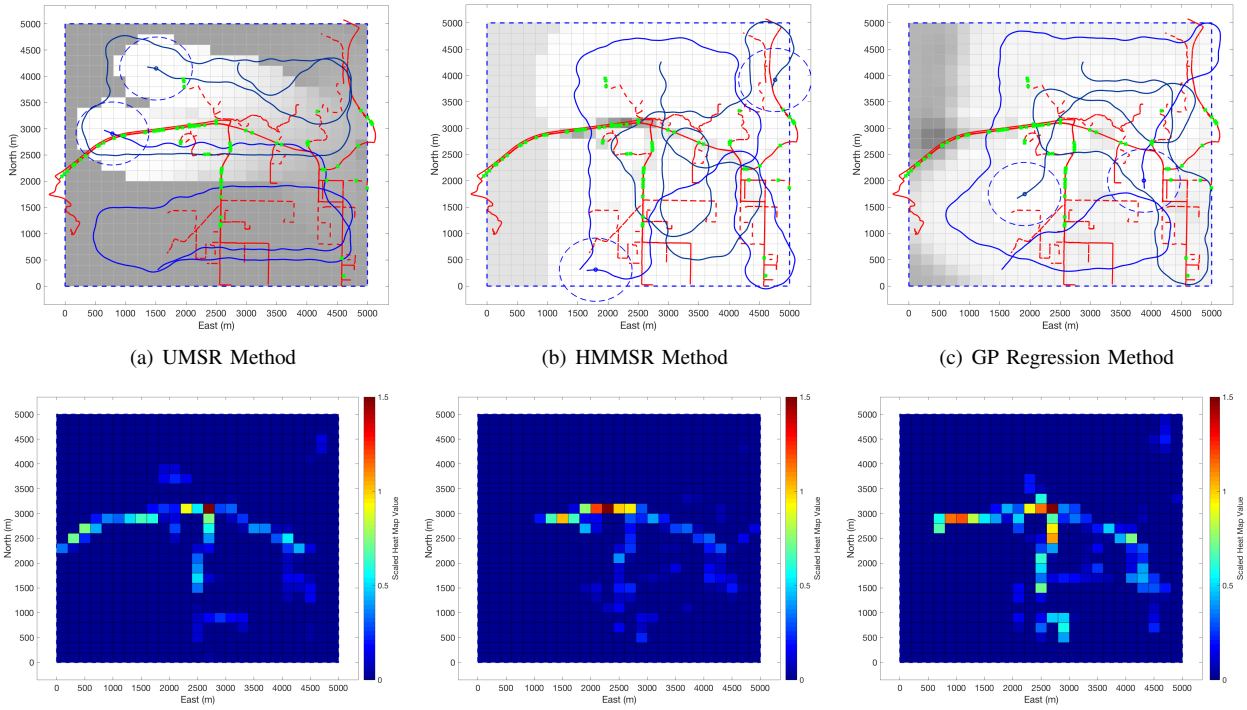
(b) Overall heat map for Chatsworth.

Fig. 4. Shows the Chatsworth simulation environment at time $t = 550$ seconds.

Each of the three search methods is tested in the Chatsworth simulation with 100 Monte Carlo runs, ten look-ahead steps, 181 total targets, and two UAVs. Fig. (5) shows simulation results of a single Monte Carlo run.

Panel 5(a), 5(b), and 5(c) show the vehicle and target paths given at $t = 550$ seconds for each of the three search algorithms. The red lines are the paths of the targets shown over the entire simulation time and the green x symbols indicate where targets are located at the final time ($t = 550$ seconds). The blue circles are the two UAVs with the larger dotted blue circles representing their sensing radii. The blue arrow is the UAV's velocity vector. The two long blue lines show the paths taken by the UAVs for the simulation time. The gray grid cells in the simulation indicate the current reward value for searching that cell. The darker the grid cell, the higher the reward. Where the UAV is currently sensing is white because the UAV has already gained that reward.

Panel 5(a) shows the simulation for the UMSR method. As expected, it is evident that the UAVs are covering the search area in an even manner and showing no discretion to target densities. The decision to revisit areas which have previously been searched is dictated by the amount of time which has passed since it was last viewed. This is seen in the grid cell rewards. Cells which were recently searched have a light gray color, while those which haven't been searched in a long time are dark gray. Panel 5(d) shows the learned heat



(a) UMSR Method (b) HMMSR Method (c) GP Regression Method

(d) Heat map for UMSR (e) Heat map for HMMSR (f) Heat map for GP regression

Fig. 5. Shows Chatsworth simulation for each of the search methods at time $t = 550$ seconds.

map from the UMSR method. In comparing it to panel 4(b), the outline of the road can be seen, but it lacks the clarity from repeated measurements.

Panel 5(b) shows the HMMSR simulation. The influence of the learned heat map is seen in the grid cell rewards. There is a darker grid cell in the center of the search area because the UAVs have learned that there tend to be many targets in that area. The freeway and main streets of Chatsworth tend to receive the highest rewards. This is because the maximum grid cell rewards, $J_{g,max}$, were set to a higher value so those grid cells are searched more often. This leads the UAVs to gravitate toward areas where there are more targets and they don't travel evenly across the search area as they did in Panel 5(a). Using the HMMSR method results in seeing more targets and having a more accurate heat map than the UMSR method, even though they may not encourage exploration as much. This can be seen from it's heat map (Panel 5(e)). It provides a closer representation of the overall heat map and has a lower RMS error than the UMSR method for the high density regions, but it lacks the comprehensive coverage given by Panel 5(d).

The simulation for our GP regression method is seen in Panel 5(c). The paths of the UAVs are similar to those in the HMMSR method in that the UAVs spend more time focused on the road networks in the south-west part of the search area. The UAVs gravitate to the areas where more targets tend to be. However, with GP regression there is clearly more exploration done by the UAVS. This is seen in the comparisons of their heat maps. Panel 5(f) shows the heat map for the GP regression method which has even more detail than the heat map for HMMSR. The freeway is almost completely captured in the heat map all the way to the west border of the search area.

One of the advantages of using a GP regression is its ability to produce predictions of target densities in areas which have not been searched. In the west-central section of the search area in panel 5(c), there is a darker spot where the UAVs have not searched. This means the GP regression, due to information gathered in neighboring areas, has predicted that there is a high average amount of targets that will be in those grid cells. Because of this high prediction, the maximum grid cell reward is assigned a high value. This will lead the UAVs to treat searching those grid cells as a higher priority than others. In the case of this simulation, the prediction of the GP regression is correct and the UAV would find a high density of targets in those cells.

Fig. (6) shows the RMS error of the heat maps for all three methods from time $t = [0, 300]$ seconds. Each of the methods start out with the same error and stay close for the first part of the simulation (around 25 seconds). But after that, the GP regression error clearly drops below the rest and stays lower for the remainder of the simulations. The HMMSR method does worse than the UMSR method for almost all of the simulation. This is rooted in the HMMSR method relying fully on exploitation of knowledge, rather than balancing with exploration in order to gain more knowledge.

Table I shows the averages and standard deviation of the mean heat map RMS error and fraction of targets seen for each search method. As expected, the GP regression method outperformed the other two methods in heat map RMS error.

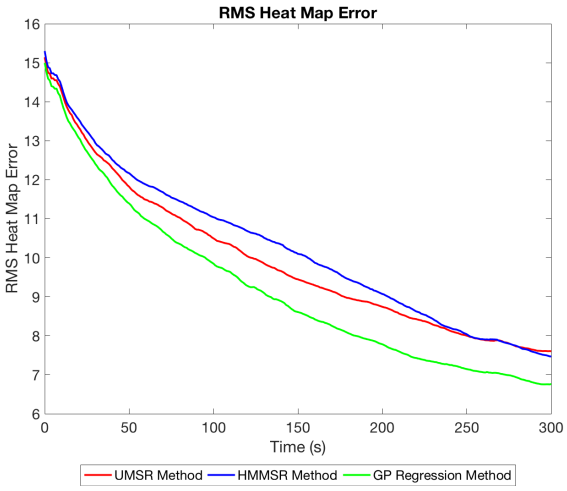


Fig. 6. Heat map RMS error over the course of the Chatsworth simulation.

TABLE I

CHATSWORTH SIMULATIONS AVERAGE AND STANDARD DEVIATION OF THE RMS ERROR. AND THE AVERAGE AND STANDARD DEVIATION OF THE AVERAGE FRACTION OF TARGETS SEEN.

Approach	Final Heat Map Error	Average % Targets Seen
UMSR	7.61+/-1.10	11.1+/-2.8%
HMMSR	7.47+/-1.24	11.4+/-3.0%
GP Regression	**6.75+/-0.94**	**13.2+/-3.0**%

It had an average error of 6.75, while UMSR and HMMSR had errors of 7.61 and 7.47 respectively. Performing a two sample t-test on the error of the GP regression method and UMSR resulted in a p-value of 9.53e-9. Doing the same with the GP regression method and HMMSR resulted in a p-value of 3.56e-6. This shows that the RMS errors between the different methods is statistical significant.

Likewise, Table I shows the GP regression method also performs better in the average fraction of targets seen, with an average of 13.2%. This surpasses HMMSR with 11.4% and UMSR with 11.1%. Performing the two sample t-test between the GP regression method and UMSR produces a p-value of 4.99e-7, and doing the same test between the GP regression method and HMMSR produces a p-value of 1.71e-5. These p-values show a statistical significance between the fraction of targets seen.

B. Sparse Search Simulation

Our second simulation environment is designed to replicate a military scenario. The search area is a 5000 meter square region in a sparsely visited area. There are two hot spots of activity representing military camps or bases which need to be found and searched. Panel 7(a) shows the search area (blue dotted box) and target paths (red lines). An overall heat map for search area is seen in panel 7(b). The two hots spots are clearly seen while all other surrounding grids have no target tracks.

All three search methods are compared with 100 Monte Carlo runs, ten look-ahead steps, 20 total targets, and two UAVs. A single Monte Carlo simulation is depicted for each method in Fig. (8).

Panel 8(a) shows the UMSR method. Again, the UAV paths traverse across the search area in an even manner without regard to target locations. Its heat map in Panel 8(d) shows a fairly representative picture compared to the overall heat map in Panel 7(b).

The HMMSR method, in Panel 8(b), led the UAVs to search the north-east corner of the area multiple times. The UAVs still explored the whole search area and didn't stay fixated on the two high density areas. The south-west corner of the search area has high grid cell reward because of the learned heat map seen in panel 8(e).

The GP regression method performed well in the sparse simulation. In Panel 8(c) it is clear that one of the UAVs started in the south-west corner where there is a high density of targets, explored north, returned to the high density corner, explored east, and returned again. This shows the balance of exploration and exploitation. The UAVs continue to explore, but return to the areas of the highest target densities. The GP regression identified and rewarded the two areas of highest target densities. The learned heat map in panel 8(f) is very comparable to the overall heat map.

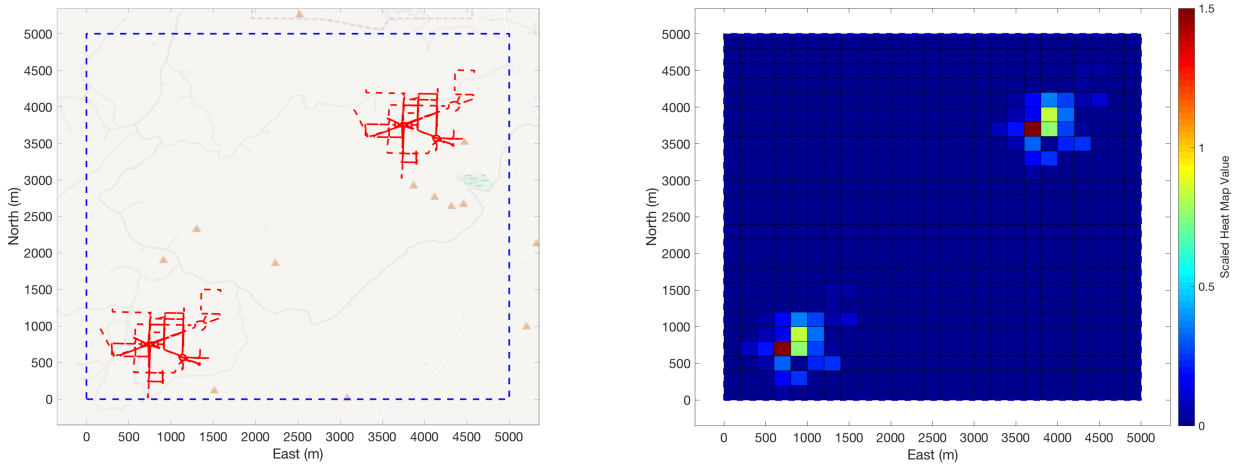
Fig. (9) shows the average heat map RMS error for each of the three methods over the course of the 100 Monte Carlo runs. For the sparse simulation, the RMS errors for each of the three methods stayed closer together and did not diverge as much, when compared with the Chatsworth simulation. At the beginning of the simulations, the GP regression matched the RMS error of the other methods closely, but as time went on it improved more than the others. In this test, the UMSR method again had a lower RMS error than the HMMSR method. The two methods had very close RMS errors throughout the simulation. This likely resulted from the search area being mostly empty, and ten look-ahead steps isn't enough for the UAVs using HMMSR to plan paths back to the areas of high target densities. Once the UAVs are in the uniformly empty space, they act similarly to the UMSR method. The HMMSR method is then slightly disadvantaged because of low uniform maximum grid cell values, which created negligible variability in grid cell values and didn't help dictate planned vehicle paths.

TABLE II

SPARSE SIMULATIONS AVERAGE AND STANDARD DEVIATION OF THE RMS ERROR. AND THE AVERAGE AND STANDARD DEVIATION OF THE AVERAGE FRACTION OF TARGETS SEEN.

Approach	Final Heat Map Error	Average % Targets Seen
UMSR	3.32+/-.40	12.9+/-3.8%
HMMSR	3.37+/-.42	13.1+/-3.9%
GP Regression	**3.25+/-.40**	**14.0+/-3.9**%

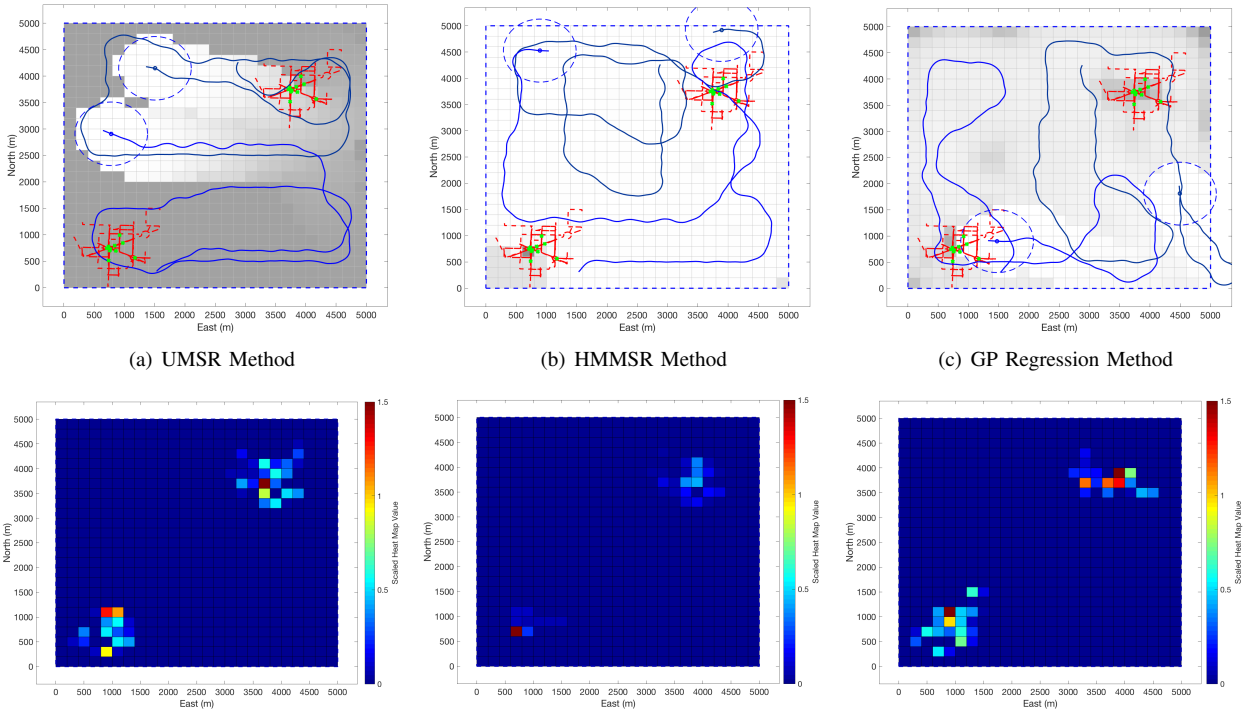
Table II contains the averages and standard deviation of the mean heat map RMS error and fraction of targets seen. Here is also shown that the GP regression method had the lowest RMS heat map error of 3.25. UMSR was next lowest with



(a) Sparse search simulation with target paths.

(b) Overall heat map for the sparse simulation.

Fig. 7. Shows the sparse simulation environment at time $t = 550$ seconds.



(a) UMSR Method

(b) HMMSR Method

(c) GP Regression Method

(d) Heat map for UMSR

(e) Heat map for HMMSR

(f) Heat map for GP regression

Fig. 8. Shows a sparse simulation for each of the search methods at time $t = 550$ seconds.

3.32 and then HMMSR with 3.37. Performing a two sample t-test on the RMS error of the GP regression method and UMSR resulted in a p-value of .117, which is not statistically significant. For the GP regression method and HMMSR, the two sample t-test resulted in a p-value of .020. Due to the large empty spaces with no targets, this simulation allows for any method to gather a nearly correct estimation of target densities over a short amount of time. Each of the methods act in similar ways until enough time has passed that the different methods can react to the learned information. For example, the GP regression has a clearer advantage near 200 seconds into the simulation, as seen in Fig. (9).

Again, the GP regression method had the highest fraction of targets seen with 14.0%. HMMSR and UMSR lower and had very close values of 13.1 and 12.9, respectively. This shows the advantage of using the GP regression as part of the algorithm, rather than relying solely on the heat map. A two sample t-test on the fraction of targets seen for the GP regression method and UMSR produces a p-value of .017. The same test for the GP regression method and HMMSR

results in a p-value of .043. These results are significant on a 95% confidence level, but are not as significant as the Chatsworth simulation. In this simulation, the number of targets was much lower and they were grouped closely together. As the targets moved in the search space, it created large changes in the heat map, allowing for an increase in the RMS error when sensed by the UAVs. This simulation shows there is a greater advantage in using the GP regression method when there is a higher number of targets and the nonuniform target densities is dispersed across many grid cells.

V. Conclusion

This paper presents an improved method for searching, patrolling, or surveillance in areas with nonuniform target densities. While searching, this GP regression method uses a continual stream of new knowledge about target locations to influence how the UAVs coordinate their paths. Larger rewards are given for searching areas with high target densities. The GP regression does well at estimating and predicting target densities, even in regions of the search area which have not yet been seen. Results from testing show the numerical advantages of using a GP regression when compared with baseline search methods. As illustrated with two example simulations, the GP regression method balanced exploitation and exploration while maintaining the highest percentage of targets tracked and lowest error in its learned heat map of the target densities.

CKP: Future work in this area will include...evasive targets...other?

ACKNOWLEDGMENT

This work has been funded by the Center for Unmanned Aircraft Systems (C-UAS), a National Science Foundation Industry/University Cooperative Research Center (I/UCRC) under NSF award No. IIP-1650547 along with significant contributions from C-UAS industry members.

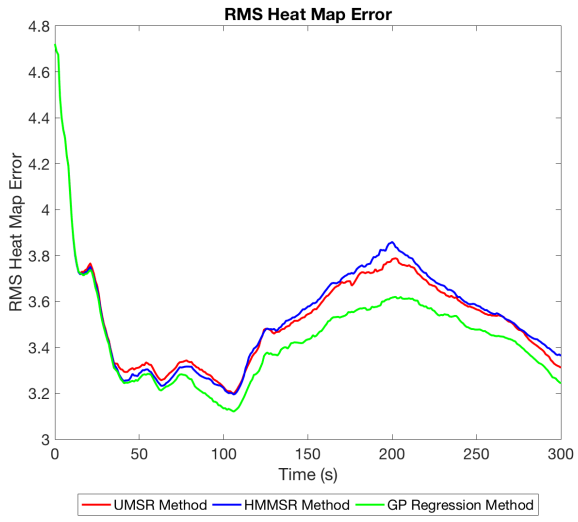


Fig. 9. Heat map RMS error over the course of the sparse simulation.

References

[1] K. S. Lee, M. Ovinis, T. Nagarajan, R. Seulin, and O. Morel, "Autonomous patrol and surveillance system using unmanned aerial vehicles," in *2015 IEEE 15th International Conference on Environment and Electrical Engineering (EEEIC)*, pp. 1291–1297, June 2015.

[2] M. Mueller, G. Sharma, N. Smith, and B. Ghanem, "Persistent aerial tracking system for uavs," *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1562–1569, Oct 2016.

[3] S. Waharte and N. Trigoni, "Supporting search and rescue operations with uavs," in *2010 International Conference on Emerging Security Technologies*, pp. 142–147, Sept 2010.

[4] D. W. Casbeer, D. B. Kingston, R. W. Beard, and T. W. McLain, "Cooperative forest fire surveillance using a team of small unmanned air vehicles," *International Journal of Systems Science*, vol. 37, no. 6, pp. 351–360, 2006.

[5] T. Suzuki, D. Miyoshi, J. Meguro, Y. Amano, T. Hashizume, K. Sato, and J. Takiguchi, "Real-time hazard map generation using small unmanned aerial vehicle," in *2008 SICE Annual Conference*, pp. 443–446, Aug 2008.

[6] X. Tian, Y. Bar-Shalom, and K. R. Pattipati, "Multi-step look-ahead policy for autonomous cooperative surveillance by uavs in hostile environments," *Proceedings of the IEEE Conference on Decision and Control*, vol. 5, no. 1, pp. 2438–2443, 2008.

[7] A. Ryan, M. Zennaro, A. Howell, R. Sengupta, and J. Hedrick, "An Overview of Emerging Results in Cooperative UAV Control," *Proceedings of 43rd IEEE Conference on Decision and Control*, 2004.

[8] L. E. Caraballo, J. J. Acevedo, J. M. Díaz-Báñez, B. C. Arrue, I. Maza, and A. Ollero, "The block-sharing strategy for area monitoring missions using a decentralized multi-uav system," in *2014 International Conference on Unmanned Aircraft Systems (ICUAS)*, pp. 602–610, May 2014.

[9] N. Nigam, "The multiple unmanned air vehicle persistent surveillance problem: A review," *Machines*, vol. 2, no. 1, pp. 13–72, 2014.

[10] Y. Elmaliach, A. Shiloni, and G. Kaminka, "A realistic model of frequency-based multi-robot polyline patrolling," *Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems*, vol. 1, pp. 63–70, 2008.

[11] R. R. Zargar, M. Sohrabi, M. Afsharchi, and S. Amani, "Decentralized area patrolling for teams of uavs," in *2016 4th International Conference on Control, Instrumentation, and Automation (ICCIA)*, pp. 475–480, January 2016.

[12] Z. Hu and D. Zhao, "Reinforcement learning for multi-agent patrol policy," in *Cognitive Informatics (ICCI), 2010 9th IEEE International Conference on*, pp. 530–535, July 2010.

[13] L. Panait and S. Luke, "Cooperative multi-agent learning: The state of the art," *Autonomous Agents and Multi-Agent Systems*, vol. 11, no. 3, pp. 387–434, 2005.

[14] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)*. The MIT Press, 2005.

[15] S. Yang, N. Wei, S. Jeon, R. Bencatel, and A. Girard, "Real-time optimal path planning and wind estimation using gaussian process regression for precision airdrop," in *2017 American Control Conference (ACC)*, pp. 2582–2587, May 2017.

[16] N. R. Lawrance and S. Sukkarieh, "Autonomous exploration of a wind field with a gliding aircraft," *Journal of Guidance, Control, and Dynamics*, vol. 34, no. 3, pp. 719–733, 2011.

[17] A. G. Wilson and R. P. Adams, "Gaussian process kernels for pattern discovery and extrapolation," *Proceedings of the 30th International Conference on Machine Learning*, vol. 28, 2013.

[18] J. Kocijan, R. Murray-Smith, C. E. Rasmussen, and A. Girard, "Gaussian process model based predictive control," *American Control Conference, 2004. Proceedings of the 2004*, vol. 3, pp. 2214–2219, 2004.

[19] G. Cao, E. M.-K. Lai, and F. Alam, "Gaussian process model predictive control of an unmanned quadrotor," *Journal of Intelligent & Robotic Systems*, vol. 88, no. 1, pp. 147–162, 2017.

[20] W. Ren, R. W. Beard, and E. M. A. Atkins, "A Survey of Consensus Problems in Multi-Agent Coordination," in *American Control Conference, 2005. Proceedings of the 2005*, (Portland, Oregon), pp. 1859–1864, IEEE, jun 2005.

[21] C. K. Peterson and D. A. Paley, "Multivehicle Coordination in an Estimated Time-Varying Flowfield," *Journal of Guidance, Control, and Dynamics*, vol. 34, no. 1, pp. 177–191, 2011.

[22] K. M. Lynch, I. B. Schwartz, P. Yang, and R. A. Freeman, "Decentralized Environmental Modeling by Mobile Sensor Networks," *IEEE Transactions on Robotics*, vol. 24, no. 3, pp. 710–724, 2008.

[23] T. Li, F. Hlawatsch, and P. M. Djuri, "Cardinality-Consensus-Based PHD Filtering for Distributed Multitarget Tracking," *IEEE Signal Processing Letters*, vol. 26, no. 1, pp. 49–53, 2019.

[24] B. G. Moon and C. K. Peterson, "Learned search parameters for cooperating vehicles using gaussian process regressions," in *2018 International Conference on Unmanned Aircraft Systems (ICUAS)*, pp. 493–502, IEEE, 2018.

[25] C. K. Peterson, "Dynamic grouping of cooperating vehicles using a receding horizon controller for ground target search and track missions," *2017 IEEE Conference on Control Technology and Applications (CCTA)*, pp. 1855–1860, 2017.

[26] A. J. Newman, S. R. Martin, J. T. DeSena, J. C. Clarke, J. W. McDerment, W. O. Preissler, and C. K. Peterson, "Receding Horizon Controller using Particle Swarm Optimization for Closed Loop Ground Target Surveillance and Tracking," *Signal Processing, Sensor Fusion, and Target Recogniton*, vol. 7336, no. 1, pp. 73360M–1–73360M–12, 2009.

[27] A. Mchutchon and C. E. Rasmussen, "Gaussian process training with input noise," *Advances in Neural Information Processing Systems*, pp. 1341–1349, 2011.

[28] J. Quinonero-Candela, A. Girard, and C. E. Rasmussen, "Prediction at an uncertain input for gaussian processes and relevance vector machines-application to multiple-step ahead time-series forecasting," tech. rep., Technical Report, IMM, Danish Technical University, 2002.

[29] D. Krajzewicz, J. Erdmann, M. Behrisch, and L. Bieker, "Recent development and applications of SUMO - Simulation of Urban MObility," *International Journal On Advances in Systems and Measurements*, vol. 5, pp. 128–138, December 2012.