Hierarchical Heuristic Search Using a Gaussian Mixture Model for UAV Coverage Planning

Lanny Lin, *Member, IEEE,* and Michael A. Goodrich, *Senior Member, IEEE*

***Abstract*—During unmanned aerial vehicle (UAV) search missions, efficient use of UAV flight time requires flight paths that maximize the probability of finding the desired subject. The probability of detecting the desired subject based on UAV sensor information can vary in different search areas due to environment elements like varying vegetation density or lighting conditions, making it likely that the UAV can only partially detect the subject. This adds another dimension of complexity to the already difficult (NP-Hard) problem of finding an optimal search path. We present a new class of algorithms that account for partial detection in the form of a task difficulty map and produce paths that approximate the payoff of optimal solutions. The algorithms use the *mode goodness ratio* heuristic that uses a Gaussian mixture model to prioritize search subregions. The algorithms search for effective paths through the parameter space at different levels of resolution. We compare the performance of the new algorithms against two published algorithms (Bourgault's algorithm and LHC-GW-CONV algorithm) in simulated searches with three real search and rescue scenarios, and show that the new algorithms outperform existing algorithms significantly and can yield efficient paths that yield payoffs near the optimal.**

***Index Terms*—Heuristic algorithms, hierarchical systems, navigation, path planning, unmanned aerial vehicles.**

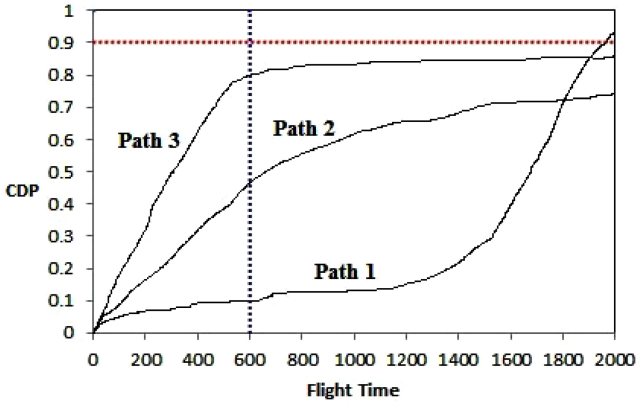

Fig. 1. Two approaches to the probability-maximizing path-planning problem. With three paths are generated by various algorithms, the first approach prefers the path maximizing the CDP given a specific flight time (path 3 is the winner) and the second approach prefers the path achieving a specified CDP in the shortest amount of time (path 1 is the winner).

I. Introduction

MINI-unmanned aerial vehicles (UAVs) are becoming useful tools in many reconnaissance, remote-sensing, surveillance, and search operations, thanks to advances in UAV technologies. They can help firefighters to map forest fires, news crews to provide coverage, police to monitor crowds, and wilderness search and rescue (WiSAR) workers to locate a missing person. In these applications, the UAV uses its on-board cameras to provide useful visual information in support of the specific operation.

This paper focuses on using mini-UAVs to support WiSAR. The aerial view from a UAV enables WiSAR workers to survey large areas of importance in real time [1]. Search efficiency is very important in WiSAR because, as time progresses, the survivability of the missing person decreases and the effective search radius increases by approximately 3 km/h [2]. Therefore, a good flight path should rapidly maximize the probability of finding the missing person to make efficient use of the limited flying time.

Each UAV path accumulates information over time as the UAV's sensors scan the ground. As illustrated in Fig. 1, various paths do so in different ways depending on how information is distributed in the environment. The goal is to maximize the total probability of detection. There are two quality metrics for the probability-maximizing path-planning problem [3]–[5]. First, find the path that maximizes the cumulated detection probability (CDP) after a specific flight time (blue vertical dotted line). Out of the three example paths in Fig. 1, path 3 becomes the winner. Second, find the path that achieves a desired CDP in the shortest amount of time (red horizontal dotted line). Path 1 would become the winner out of the three, instead. We model the problem following the first approach.

When using a UAV's on-board camera to assist WiSAR operations, factors such as dense vegetation, lighting conditions, shadows, or distance between the camera and the ground can lower the quality of the UAV aerial view and decrease the probability of detection [6]. This can be attributed to both sensor and human limitations (such as limited attention span and cognitive workload). We propose to represent partial

Manuscript received March 8, 2013; revised December 1, 2013 and February 19, 2014; accepted February 21, 2014. This work was supported in part by the National Science Foundation under Grant 0534736 and in part by a grant from the Army Research Laboratory. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the sponsoring organizations.This paper was recommended by Associate Editor A. Tayebi.

The authors are with the Department of Computer Science, Brigham Young University, Provo, UT 84602 USA (e-mail: lanny.lin@byu.edu; mike@cs.byu.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCYB.2014.2309898

detection in the form of a task difficulty map, where a more difficult subregion on the map has lower probability of detection. Using a task difficulty map enables us to integrate geo-referenced and spatial-related sensor constraints into the problem formulation, which supplements traditional sensor modeling methods (e.g., [7]) and potentially improves search performance in real-world search scenarios. Because detection difficulties vary in different search subregions, flying patterns such as lawnmower and Zamboni do not guarantee optimal coverage. Integrating the task difficulty map into path planning adds another dimension of complexity to the already difficult problem and causes the performance of existing greedy-type algorithms [Bourgault's algorithm (BA) [7] and LHC-GW-CONV algorithm [8]] to suffer.

We model the path-planning problem as a discrete combinatorial optimization problem and propose a new heuristic, the mode goodness (MG) ratio. This heuristic uses a Gaussian mixture model (GMM) to identify and prioritize search subregions. We then present two new algorithms (*Top2* and *TopN*) that utilize the heuristic in hierarchical path planning by forcing the UAV to visit high-priority subregions. The hierarchical structure enables the algorithms to: 1) cluster probability volumes and 2) prioritize search subregions at different levels of resolution. It also makes it easy to parallelize the two new algorithms and improve computation speed. We compare the performance of the new algorithms against two published algorithms (BA [7] and LHC-GW-CONV algorithm [8]) in simulated searches based on three real search and rescue (SAR) scenarios. Results show that the new algorithms outperform existing algorithms significantly and can yield efficient paths that approximate the payoff of the optimal path.

The contributions of this paper are: 1) the introduction of GMM to compute the mode good ratio (MGR) heuristic, which can be used to prioritize search subregions in a hierarchical planner; 2) two new path-planning algorithms that utilize the MGR heuristic to improve path-planning performance; and 3) the use of a spatial representation (task difficulty map) in modeling sensor detection probability with terrain and vegetation information and incorporating that into UAV path planning.

Section II defines the problem and the metrics used to evaluate algorithm performance. Section III discusses related literature. Section IV reviews two existing algorithms and then demonstrates the weakness of these algorithms with a synthetic scenario. It then presents the MGR heuristic and the two new algorithms (*Top2* and *TopN*). Section V compares algorithm performance with three real SAR scenarios. Section VI discusses the limitations of the approach. Section VII presents the summary.

II. Problem Formulation

A. Problem Framework

Typical UAVs (fixed-wing or rotorcraft) are highly mobile and variable, but we will assume a set of useful constraints on their capabilities: they have a gimbaled camera, can maintain a constant height above ground, and can travel at constant speed. A gimbaled camera enables the camera to aim straight down even when the UAV is performing roll or yaw maneuvers. We assume that the UAV's speed is much higher than the speed of the missing person and treat the missing person as stationary. At every UAV flight time step, we treat the camera footprint of the search area as a glimpse. This way we can discretize the search area, model the UAV path-planning problem as a discrete combinatorial optimization problem with respect to probability accumulated, and define it following the framework described in [9].

The search space is represented as a finite, connected graph $G = (V, E)$. V denotes the set $\{v_1, ..., v_n\}$ as vertices of G and E denotes the set of edges. Each edge in E can be viewed as an unordered pair of vertices $\{v_i, v_j\}$. The missing person is located at one of the vertices of G. A given probability distribution map for the missing person is discretized to match graph G, with p_i being the probability that the missing person is located at vertex v_i. It is obvious that

$$\sum_{i=1}^{n} p_i = 1. \tag{1}$$

The UAV search is conducted in discrete time. During each time step, the UAV camera footprint can cover one vertex. For a desired flight with T time steps, let S denote the set $\{0, 1, 2, ..., T\}$. The UAV's motion is constrained by the structure of the graph G. Let Ψ be the set of functions $\psi : S \mapsto V$ with the property that for any two consecutive integers t and $t+1$ in S, either $\psi(t) = \psi(t+1)$ or $\{\psi(t), \psi(t+1)\} \in E$. Here, Ψ represents all possible paths, and under path ψ, vertex $\psi(t)$ is searched during step t. The conditions on the set Ψ guarantee that at each time step, the UAV camera footprint will either remain at the current vertex (only possible for a rotorcraft) or move to a neighboring vertex.

Even when the UAV camera footprint covers the vertex occupied by the missing person, it is not certain that a detection will occur. The probability of detection is described by a glimpse probability function g, which is defined by a given task difficulty map. The task difficulty map is a spatial representation of sensor detection probability and defines areas where it is difficult to detect the missing person (with lower probability of detection), with d_i being the task difficulty level at vertex v_i. Let $d_{\max}$ be the maximum task difficulty level in the given map. If at time step t the UAV camera footprint covers vertex v, then let $g(v, t)$ be the probability that a detection will occur, given that the missing person is at vertex v. We model this as

$$g(v_i, t) = 1 - \frac{d_i}{d_{\max} + 1} \tag{2}$$

so that more difficult tasks (higher d_i values) have lower glimpse detection values, g.

Let $P_T(\psi)$ represent the CDP for path $\psi \in \Psi$ with T time steps. For each (cell, time) pair (i, t) with $1 \leq i \leq n$ and $0 \leq t \leq T$, we define the probability of failure $f(i, t, \psi)$ by

$$f(i, t, \psi) = \begin{cases} 1 - g(v_i, t) & \text{if } \psi(t) = v_i \\ 1 & \text{otherwise.} \end{cases} \tag{3}$$

Let D_j represent a detection on the jth observation so that $\overline{D}_j$ is a detection failure. Then the probability of failing to detect the missing person after N observations of vertex v_i, given that the missing person is at vertex v_i, is the joint probability $P(\overline{D}_1, \overline{D}_2, ..., \overline{D}_N|v_i)$. Assuming each observation is conditionally independent of each other (typical in the WiSAR literature), we can rewrite the joint probability as

$$P(\overline{D}_{1:N}|v_i) = \prod_{j=1}^{N} P(\overline{D}_j|v_i) \tag{4}$$

and the probability of detecting the missing person after N observations as

$$P(D_{1:N}|v_i) = 1 - P(\overline{D}_{1:N}|v_i) \tag{5}$$

which is equivalent to

$$P(D_{1:N}|v_i) = 1 - \prod_{t=0}^{T} f(i, t, \psi) \tag{6}$$

where N is how many times v_i shows in path ψ. Then $P_T(\psi)$ can be computed by

$$P_T(\psi) = \sum_{i=1}^{n} p_i \Big(1 - \prod_{t=0}^{T} f(i, t, \psi)\Big) \tag{7}$$

where p_i is the probability that the missing person is located at vertex v_i. Define $\exists\psi^*\in\Psi$ such that for any alternate path $\psi'\in\Psi$, $P_T(\psi^*)\geq P_T(\psi')$. Our goal is to find the optimal path ψ^* that produces the maximum CDP (path 3 in Fig. 1 at $T = 600$ if there are only three possible paths) or find an efficient path ψ' that produces payoff approximating the payoff of the optimal path within reasonable computation time.

When the path needs to end at an operator-specified vertex (for easy UAV retrieval or to join with other path segments), we add the constraint to the problem formulation (only add edges to a path that does not violate the constraint so the UAV that has enough time to reach the end point). Also for fixed-wing UAVs, additional motion constraints (such as not allowing the UAV to fly backward) are also introduced as velocity constraints affecting edges $\{\psi(t), \psi(t_1)\}$, effectively creating a directed graph. Both proposed path-planning algorithms satisfy these constraints.

B. Performance Metrics

We will use two measures of algorithm performance: the quality of the path and the run-time of the algorithm. Run-time is self-explanatory, but we need a measure of path quality. Ideally, the efficiency of an algorithm should be computed with the following equation:

$$Efficiency(\psi') = \frac{P_T(\psi')}{P_T(\psi^*)} \tag{8}$$

where ψ^* is the optimal path. However, because we do not know what ψ^* is, we bound the efficiency. Let ψ_{teleport} be defined as the path constructed as follows: 1) deduct the time needed to move from the start vertex to the nearest vertex with nonzero p_i (plus time for doing the same with the end vertex if specified) and 2) at each step, the UAV teleports to the vertex that allows the UAV to collect the highest amount of probability after considering the task difficulty at that vertex. Then, all the probability collected during the teleport flight is summed, giving $\text{Efficiency}_{\text{LB}_i}$ for path i

$$rmEfficiency_{\text{LB}_i} = \frac{P_T(\psi_i)}{P_T(\psi_{rmteleport})}. \tag{9}$$

Since $P_T(\psi^*) \leq P_T(\psi_{\text{teleport}})$, Efficiency can be no worse than $\text{Efficiency}_{\text{LB}}$, the latter sets a lower bound for the true efficiency. Note that the majority of the teleport path ψ_{teleport} is made up of disjointed points because the UAV would be "jumping" from vertex to vertex, always landing on the vertex that promises highest amount of probability collectable.

III. Related Work

Many path-planning algorithms in the literature address obstacle avoidance while planning a path to reach a destination using A* [10], D* [11], Voroni diagrams [12], or probability roadmaps and rapidly exploring random trees [13]. Hierarchical heuristics approaches were also developed, such as hierarchical A* (HA*) by Holte *et al*. [14], hierarchical task-based real-time path planning by Meuleau and Brafman [15], and hierarchical-AO* (HiAO*) by Naveed *et al*. [16]. The algorithms we present solve a different path-planning problem by generating paths that make efficient use of the limited travel time and maximizing the probability of finding the missing person. This is similar to the vehicle routing problem [17] and the orienteering problem (OP), which is a variation of the traveling salesman problem (TSP) and is known to be NP-Hard [18]. However, our path-planning problem is even more difficult with added challenges of repeated visits and partial detection. Tasgetiren and Smith [19] proposed a genetic algorithm to solve OP. Liang and Smith [20] presented an ant colony optimization approach that uses an unusual sequenced local search and a distance-based penalty function for path planning. These algorithms work well with OP problems with few nodes (21–100) but can be slow with many nodes. Unfortunately, they do not allow repeated visits and do not support partial detection. Although the classic dynamic programming [21] method can solve TSP, because TSP is NP-Hard, it cannot be solved in polynomial time, unless $P = \text{NP}$. The method suffers the "curse of dimensionality" and does not scale well with complex problems. Reinforcement learning (approximate dynamic programming) methods [22], [23] seek approximate solution (suboptimal policies) using function approximation, thus avoiding the "curse of dimensionality" and scaling better. However, in our path-planning problem, because a node can be visited multiple times, and our Bayesian approach that allows for partial collection of information, the score/prize collected for each visit is different. When using a reinforcement learning method, the reward and the value functions both become path dependent, making the design of a reinforcement learning approach more challenging. We chose a heuristic approach that scales well when search area and flight duration expand.

In the 1950s, Koopman [24] discussed the uncertainties in the act of detecting hostile submarines with radars and

proposed a concept called the instantaneous probability of detection by one glimpse. He presented simple search algorithms and demonstrated how search effort should be distributed given a prior probability distribution of the target and known law of detection when only a limited total amount of search effort (or time) is available [3]. Stone [4] presents various search plans with partial detection models using Lagrange multipliers and maximization of Lagrangians in finding stationary target in very basic search problems when no false targets are present. Washburn [5] discussed how to construct optimal search paths for different search problems. The author also developed detection models based on radar/sonar and expanded the fundamentals of search theory to include moving targets. Bourgault *et al*. [7] described a Bayesian framework for UAV trajectory planning to maximize the chances of finding the target in given restricted time. Partial detection was modeled based on a downward-looking millimeter wave radar, and a one-step lookahead method was used for path planning using posterior distributions obtained from Bayes filter [25] updates. More recent work includes [26], where Niedfeldt *et al*. presented a UAV path-planning algorithm that utilizes probability of detection and maximizes the probability of identifying an object using a N-step lookahead method, and [27], where Ryan and Hedrick developed a control formulation for a fixed-wing UAV that minimizes the entropy of an estimate distribution over a receding horizon for searching a moving target. N-step lookahead and receding horizon methods are greedy-type algorithms that run into scalability bounds and generate suboptimal paths in situations when a complicated detection model is used, such as a task difficulty map.

Koester [28] compiled statistics from large set of past WiSAR incidents. These statistics can be used to construct probability distribution maps. Ferguson [29] described how Geographic Information Systems (GIS)can be used to segment search areas into probability subregions. Goodrich *et al*. [1] described how a probability distribution of likely places to find the missing person can be useful for UAV path planning. Lin and Goodrich [30] proposed a Bayesian model to create such a distribution based on terrain features and past human behavior data. The model has been evaluated using real SAR scenarios at George Mason University's MapScore web portal [31] and performed well compared to other statistical models. Stone *et al*. [32] used posterior probability maps and successfully located the wreckage of Air France Flight 447 [32]. Metrics such as Koopman's instantaneous probability of detection by one glimpse [24], "seeability" proposed by Morse *et al*. [6], and terrain and vegetation information obtained from USGS [30] can be used to build a task difficulty map representing probability of detection in different search subregions.

The MGR heuristic is used to evaluate the "peakedness" of a bivariate Gaussian. The traditional way to evaluate the peakedness of a distribution uses kurtosis [33]. Mardia [34] extended the concept to multivariate distributions. Because multivariate kurtosis is difficult to compute and may show inconsistency in meaning the peakedness of a distribution, Khurshid *et al*. [35] extended Horn's measure of peakedness [36] into a measure for bivariate normal distributions. The heuristic we propose is an even simpler method to measure the peakedness and is well adapted to support hierarchical search algorithms.

IV. Path-Planning Algorithms

In this section, we review two existing path-planning algorithms, BA [7] and LHC-GW-CONV [8], and demonstrate the weakness of these algorithms with a synthetic scenario. Next, we formally define the MGR heuristic. Then we present the Top2 and TopN algorithms.

A. BA and LHC-GW-CONV Algorithms Review

The BA algorithm [7] is a Bayesian approach to the UAV path-planning problem. Given a prior probability distribution of the missing person, it uses the Bayes filter as described in [25] to compute the posterior probability distribution at every time step. The probability of detection follows an active model of a downward-looking millimeter wave radar where signal power is determined by factors such as emitted power, antennae footprint, and sensor distance to the target. Distributions are discretized into a grid for calculation and a (greedy) one-step lookahead method is used to determine which cell the UAV should fly to next (the grid cell with the highest posterior probability).

Our formulation in Section II can be viewed as a Bayes filter with the following assumptions: p_i is the prior probability, $g(v_i, t)$ is the detection likelihood, and we assume a stationary object of interest. Instead of using a greedy approach, we look ahead much further down the path. In order to address the increased computational complexity, we use a heuristic (introduced in the next section) and rely on a hierarchical approach to improve the search for efficient paths. Because the speed of our algorithms is very fast, our algorithms can turn into a "greedy" algorithm with extended horizon when dealing with moving object or changing environment.

The sensor model in BA uses target distance and signal strength, which implicitly considers the spatial information of the environment. We use a task difficulty map to take advantage of explicit prior knowledge of the environment and how it affects the detection probability spatially. For a fair comparison, we used the same downward-looking camera visual sensor model when we implemented the BA algorithm, and we note that our algorithm can use detection models similar to the one used in [7].

The LHC-GW-CONV algorithm [8] is a combinatorial optimization approach to the UAV path-planning problem. It discretizes the given probability distribution of the missing person and the task difficulty map into a grid and uses a local hill-climbing algorithm to select the next cell to fly to (the grid cell with the highest one glimpse detection probability). Spatial averaging is performed by convolving the combined probability distribution and the task difficulty map using box filters. This serves as the tie-breaker, enabling the algorithm to look beyond local neighbors in order to plan paths toward broader areas with high probability. Even with spatial smoothing, a typical problem of Local Hill Climbing (LHC) is that it favors local maxima, resulting in the UAV getting stuck in a local probability hill for too long before it can move to

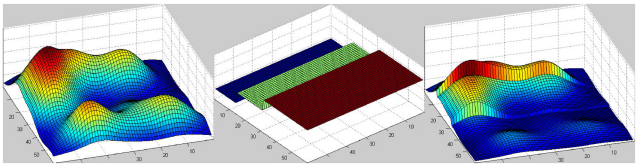

Fig. 2. Synthetic WiSAR scenario. (a) Multimodal probability distribution. (b) Simple task difficulty map. (c) Probability collectible on first visit (combining probability distribution and task difficulty map).

another probability hill. To overcome the problem, a "Global Warming" technique is used.[1] After each "ocean rise," a new path is created and the best path is returned as the final path found. Equation (10) shows how the probability p_i that the missing person has at vertex v_i changes when the ocean rises

$$p_i' \leftarrow \begin{cases} p_i - nC, & \text{if } \; p_i > nC \\ 0, & \text{otherwise} \end{cases} \tag{10}$$

where C is a constant height of each "ocean rise" and n is the number of times the "ocean" will rise.

Both BA and LHC-GW-CONV are greedy algorithms. The advantages of greedy algorithms include low computational cost and flexibility in quickly adapting to changes (e.g., a changing environment or a moving target). A major drawback is that such algorithms tend to get stuck in local maxima. We demonstrate this using the synthetic scenario in Fig. 2, which shows a multimodal distribution of the missing person location and a task difficulty map with three difficulty levels.

For a UAV path where $T = 900$, if the UAV starts from a subregion with low task difficulty (upper left corner), the BA algorithm achieved 65.99% $\text{Efficiency}_{\text{LB}}$ and the LHC-GW-CONV algorithm achieved 96.28% (averaged for 10 runs); Fig. 3 shows the paths generated by the two algorithms. The BA algorithm's performance is okay but not great, while the LHC-GW-CONV algorithm performed really well (actually slightly better than the performance of the Top2 and TopN algorithms, which we will discuss in detail in Section VI). But if the UAV starts from a subregion with high task difficulty (lower right corner), both algorithms perform poorly (much worse than the performance of the Top2 and TopN algorithms), with BA scoring 41.91% and LHC-GW-CONV scoring 53.71% (averaged for 10 runs) in $\text{Efficiency}_{\text{LB}}$; Fig. 4 shows the paths generated by the two algorithms. This is because both greedy algorithms fail to move the UAV quickly out of the local probability hill. The Top2 and TopN algorithms we propose address this problem by forcing the UAV to visit other search subregions and also allocate more flight time to subregions where the UAV can be more efficient. In order to identify better subregions, we propose the MGR heuristic.

B. Mode Goodness Ratio

The MGR heuristic prioritizes search subregions, where each subregion represents a cluster of probability volume that can be "collected" by the UAV sensor. Compute the heuristic

[1]The name "Global Warming" comes from the metaphor where the "ocean surface" represents all the grid cells with zero probability and the "islands" represent probability hills with nonzero grid cells; as the "ocean" rises, the volume of probability hills above the water decreases.

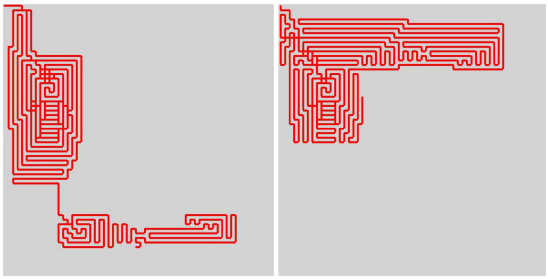

Fig. 3. Paths found at $T = 900$ when the UAV starts from a subregion with low task difficulty (upper left corner). (a) Path created by BA. (b) Path created by LHC-GW-CONV.

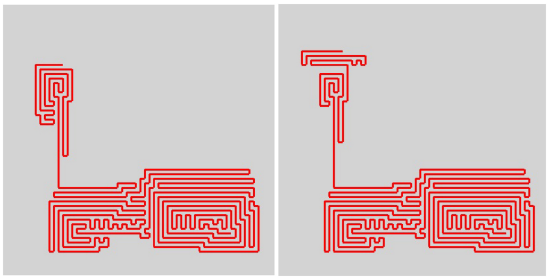

Fig. 4. Paths found at $T = 900$ when the UAV starts from a subregion with high task difficulty (lower right corner). (a) Path created by BA. (b) Path created by LHC-GW-CONV.

as follows. First, combine the probability distribution map and the task difficulty map to construct a new grid/surface G'. The value of each cell in G' represents how much probability can be collected the first time the cell is visited [e.g., Fig. 2(c)]. Second, use a GMM to partition G' into high-quality clusters/subregions. We subjectively set the maximum number of subregions to 5 to reduce computational complexity.

A GMM is a probabilistic model for finding subpopulations within an overall population and is often used for data clustering. We choose the GMM method for two reasons. 1) We can take advantage of the resulting Gaussian parameters and coefficients to estimate the peakedness of the probability hills. 2) A GMM is a parametric method, so we can define subregions by cluster probability volume hierarchically and search through the parameter space.

It is important to point out that when a task difficulty map (especially a complicated one) is applied, the resulting grid/surface G' is unlikely to resemble a mixture of Gaussians and we only use GMM to approximate the probability hills.

We used the Accord.MachineLearning library in the Accord.NET framework[2] to estimate GMM parameters. We generate data points to approximate G' (create a 2-D histogram of G' and generate number of points proportional to each bin count) and then feed these points to the Accord library, which first uses the K-means algorithm to generate k initial clusters and then uses the expectation maximization (EM) algorithm to iteratively fit data to a mixture of Gaussians. Gupta and Chen [37] provide detailed description on how to use EM to learn a GMM model. The results are a set of (k)-scaled bivariate Gaussian distributions with their means, covariance

[2]Available at: http://code.google.com/p/accord/

matrices, and the coefficient (scale) for each Gaussian. For completeness, (11) shows the density function for a multivariate Gaussian distribution

$$P(\mathbf{x}) = \frac{1}{\sqrt{(2\pi)^n |\Sigma|}} e^{-\frac{1}{2}((\mathbf{x}-\mu)^\top \Sigma^{-1}(\mathbf{x}-\mu))}. \tag{11}$$

Next, we identify all modes in the grid/surface G' (using simple local hill climbing with verification for plateaus and ridges), match the mean of each Gaussian to the closest mode centroid (in case the mode has a flat peak), and then use that centroid, C_i, to represent the subregion. Note that the number of modes in G' can be more than the number of modes in the probability distribution after a task difficulty map is applied. If there are fewer than five modes in G', we reduce k accordingly to reduce computation.

We evaluate three factors when computing MG, MG_i, for subregion i: distance ratio D_i, probability volume V_i, and subregion area A_i.

The first factor, the distance ratio, D_i, is defined as

$$D_i = \log\left(\frac{T}{\alpha_i + 1}\right) \tag{12}$$

where T is the total UAV flight time (in time steps) and α_i is the L1 norm distance from the start location of the path to the centroid of the subregion, C_i. If an end location is specified for the path, then that distance is also added

$$\alpha_i = \begin{cases} ||\mathrm{Start} - C_i||_1, & \text{no End} \\ ||\mathrm{Start} - C_i||_1 + ||\mathrm{End} - C_i||_1, & \text{otherwise.} \end{cases} \tag{13}$$

We add 1 to the denominator in (12) to make sure it will never be 0, and use the log scale to reduce wide-ranging quantities to a smaller range.

The idea behind the distance ratio is that a subregion is less attractive when it takes a large percentage of the total flight time to reach the center of the subregion because the trip to get there might not be very efficient. Therefore, higher D_i values indicate closer subregions.

The second factor, the probability volume, V_i, is defined as

$$V_i = V_{3\sigma_{x_i'} 3\sigma_{y_i'}} w_i \tag{14}$$

where $V_{3\sigma_{x_i'} 3\sigma_{y_i'}}$ is a constant (roughly 99.46%) representing the volume of probability under a standard bivariate Gaussian surface within three standard deviations, and w_i is the weight of each Gaussian component G_i, which is the coefficient of the Gaussian in the mixture as shown below with the property of $\sum_{i=1}^{k} w_i = 1$

$$p(\mathbf{x}) = \sum_{i=1}^{k} w_i G_i. \tag{15}$$

The idea behind the probability volume is that a subregion is more attractive when the volume of probability within the subregion is high, meaning visiting the subregion has the potential of collecting a large amount of probability. Therefore, higher V_i values indicate subregions with more probability.

After rotating the axes of the bivariate Gaussian to align with the eigenvectors of the covariance matrix Σ_i, the area under the surface within three standard deviations in both axes can be estimated using a rectangle with width $3\sigma_{y_i'}$ and height $3\sigma_{x_i'}$, where $\sigma_{x_i'}$ and $\sigma_{y_i'}$ are the square roots of the eigenvalues of the Gaussian's covariance matrix. The area of the rectangle A_i is the third factor in the heuristic. A larger A_i means it takes more time steps for a UAV to cover the area. Therefore, the lower A_i is, the better the subregion

$$A_i = (3\sigma_{x_i'})(3\sigma_{y_i'}) = 9\sigma_{x_i'}\sigma_{y_i'}. \tag{16}$$

When we divide V_i by A_i, we are basically estimating the peakedness of the Gaussian. Then assuming the peakedness is independent of the distance ratio D_i, we can multiply them together to compute the MG of the subregion i

$$\mathrm{MG}_i = D_i V_i {A_i}^{-1}. \tag{17}$$

Since all we really care about is the priority order of the search subregions, we can simplify computation by computing the MGR, MGR_i, for subregion i with respect to subregion 1 as the following:

$$\mathrm{MGR}_i = \frac{\mathrm{MG}_i}{\mathrm{MG}_1} = \frac{D_i V_i {A_i}^{-1}}{D_1 V_1 {A_1}^{-1}} \tag{18}$$

$$= \frac{D_i V_{3\Sigma} S_i (9\sigma_{x_i'}\sigma_{y_i'})^{-1}}{D_1 V_{3\Sigma} S_1 (9\sigma_{x_1'}\sigma_{y_1'})^{-1}} \tag{19}$$

$$= \frac{D_i S_i (\sigma_{x_i'}\sigma_{y_i'})^{-1}}{D_1 S_1 (\sigma_{x_1'}\sigma_{y_1'})^{-1}}. \tag{20}$$

Naturally, MGR_1, the MGR for subregion 1 with respect to subregion 1 will always be 1 and MGR_i for other subregions can be less or greater than 1. By sorting the MGRs of all the subregions, we have a way of prioritizing them according to their MG.

C. Top2 Algorithm

The Top2 algorithm is designed to generate paths that force the UAV to visit the top 2 subregions in the search area. This way, the heuristic-based path planner can escape from a probability hill where task difficulty is high and probability of detection is low. First, the MGR heuristic is used to identify the top 2 search subregions (represented by centroids). Then, local hill climbing is used to create the shortest path segment from the start location to the nearest centroid. If an end location is specified in the path-planning request, another path segment is created similarly from the end location to the other centroid.

The algorithm then identifies a point (vertex) equidistant from the two centroids (the green square) and launches two path-planning tasks to plan path segments from each centroid to that point using local hill climbing. By allocating different percentages of the remaining flight time to these two path-planning tasks, the Top2 algorithm can effectively search within a new dimension of time allocation. The subregion with more flight time allocated ends up with a longer path segment. Note that it is possible for the path to cover other subregions (other than the top 2) when a lot of flight time is allocated. Fig. 5 shows three time allocation examples.

A coarse-to-fine search is performed starting from a low resolution (large chunks of flight time transferred from one path-planning task to the other) and gradually increasing the

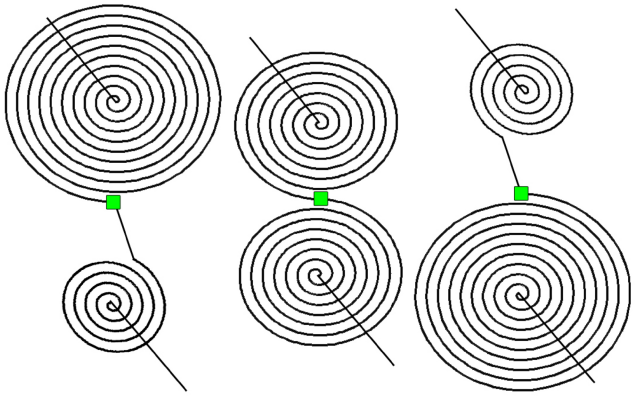

Fig. 5. Illustrations of the Top2 algorithms where the top is subregion 1 and the bottom is subregion 2. (a) More flight time allocated to subregion 1. (b) Equal flight time allocated to both subregions 1 and 2. (c) More flight time allocated to subregion 2.

```
Path Top2(Point start, int T, ArrayList centroids, Probabili-
tyMap map, int tChunk) {
1. Find closest centroid to start c1 and time needed t1
2. Plan straight path from start to c1 and store in path1
3. Find point center equidistant from c1 and c2
    map.VacuumProbability(path1);
    t2min = L1dist(c1, center);
    t3min = L1dist(c2, center);
    double efficiency = 0;
    int t2 = T-t1-t3min;
    int t3 = T-t1-t2;
    while (t2 ≥ t2min) {
        t2 -= tChunk;
        t3 += tChunk;
        (e2, path2) = LHC(c1, center, t2);
        (e3, path3) = LHC(c2, center, t3);
        if (e2 + e3 > efficiency) {
            efficiency = e2 + e3;
            pathRest = JoinPaths(path2, path3)
        }
    }
    return JoinPaths(path1, pathRest);
}
```

Fig. 6. Pseudocode for the Top2 algorithm when no end point is specified at one layer of the hierarchy (e.g., top 2 Gaussians out of 5) and one coarse-to-fine level defined by tChunk.

resolution (smaller chunks) until the best path is found. Then the path segments are joined together to form a full flight path. Fig. 6 shows the pseudocode for the Top2 algorithm.

Because we can specify how many Gaussians to fit during the GMM step, we can cluster the probability hills hierarchically; this structure enables us to search through different hierarchy layers with different k values (e.g., top 2 out of 5, top 2 out of 4, etc.). These path-planning tasks at different layers can each run the Top2 algorithm in parallel, taking advantage of the computing power of a multiprocessor system; the path with the best performance is returned as the final result.

D. TopN Algorithm

The TopN algorithm forces the UAV to visit N subregions ($5 \geq N > 1$). The algorithm first selects the top N search subregions using the MGR heuristic. Then, similar to the Top2 algorithm, it plans the two shortest path segments connecting the start and end locations of the path with the nearest centroids (modes A and D, respectively). Next, the algorithm

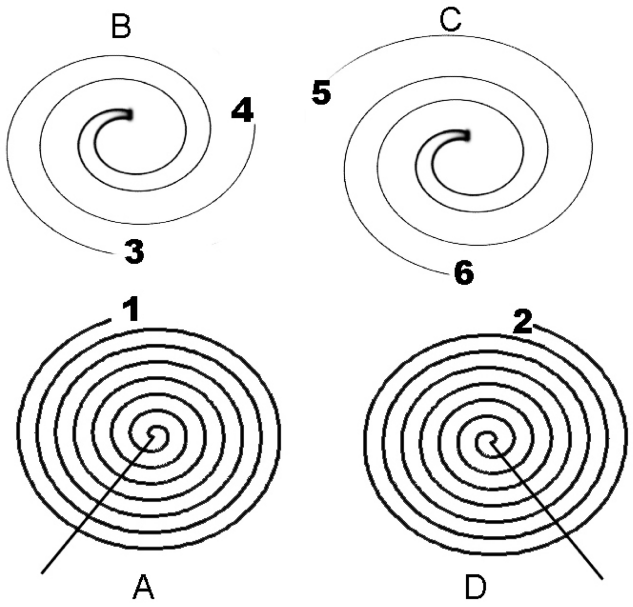

Fig. 7. Illustrations of the TopN algorithms with top 4 subregions (k = 5 and N = 4).

starts multiple path segments from the N centroids as shown in Fig. 7 (N = 4 in this example), one from the centroid nearest to the start (segment 1 from mode A), one from the centroid nearest to the end (segment 2 from mode D), and two segments for each other centroid (segment 3–6 in modes B and C). Segments 3 and 4 are connected at the center of mode B and segments 5 and 6 are connected at the center of mode C. The four segments spiral outward from the center. This technique allows the UAV to fly to the desired centroid in a "spiral in" fashion and then leave the centroid in a "spiral out" fashion without any overlaps, thus heuristically minimizing unnecessary revisits and still providing a good coverage of the probability hill. Six path segments perform local hill climbing at the same time and at each one-step lookahead, only the path segment with the maximum gain gets to add the neighboring vertex to the path. This process continues until the remaining flight time is just enough to connect all six segments in the shortest way possible. In the last step, path segments are connected into one continuous path using local hill climbing. In the example shown, segments 3 and 1 join to connect modes A and B; similarly, segments 4 and 5 connect modes B and C and segments 6 and 2 connect modes C and D. Note that planning two path segments from the center of the same Gaussian mode allows the UAV to spiral in to the center of the mode and then spiral out without crossing paths and revisit nodes, approximating a Fermat's spiral (a special type of Archimedean spiral), and improve the search efficiency (especially for an area with relatively uniform detection probability). Fig. 8 shows the pseudocode for the TopN algorithm.

Similar to the Top2 algorithm, the algorithm can specify how many Gaussians to fit during the GMM step and, in addition, search through different N values (e.g., 4 out of 5, 3 out of 5, 2 out of 5, and so on). The TopN algorithm for each hierarchy layer is run in parallel and returns the path with the best performance as the final result.

```
Path TopN(Point start, Point end, int T, ArrayList centroids,
ProbabilityMap map) {
1. Find closest centroid to start c1 and time needed t1
2. Remove c1 from ArrayList centroids
3. Plan straight path from start to c1 and store in path1
     map.VacuumProbability(path1);
4. Find closest centroid to end cN and time needed tN
5. Remove cN from ArrayList centroids
6. Plan straight path from end to cN and store in path2N
     map.VacuumProbability(path2N);
     int TLeft = T - t1 - tN;
     Path path2 = new Path();
     Path path2.add(BestNeighbor(c1));
     Path path2NMinus1 = new Path();
     Path path2NMinus1.add(BestNeighbor(cN));
     ArrayList segments = new ArrayList();
     ArrayList segments.add(path2);
     ArrayList segments.add(path2NMinus1);
     foreach (Point c in centroids) {
          Path p1 = new path();
          p1.add(c);
          Path p2 = new path();
          p2.add(BestNeighbor(c));
          segments.add(p1);
          segments.add(p2);
     }
     while (EnoughTimeToJoinAllSegments(TLeft)) {
          Path path = SegmentWithBestNeighbor(segments);
          Point p = BestNeighbor(p.lastPoint());
          path.add(p);
          map.VacuumProbability(p);
          TLeft–;
     }
     return JoinPaths(path1, path2N, segments);
}
```

Fig. 8. Pseudocode for the TopN algorithm with end point specified at one layer of the hierarchy (e.g., top 4 Gaussians out of 5).

Although the Top2 algorithm might appear similar to a special case of the TopN algorithm, where $N = 2$, it is not. First, the Top2 algorithm would force a path to go through the vertex (the green square in Fig. 5) equidistant from the two centroids; the TopN algorithm does not have this constraint. Second, although both algorithms would plan two path segments and join them together to form the final path, Top2 algorithm actually generates multiple final paths (by allocating different portion of flight time to the two path segments) at the current hierarchy and then searches for the one with the best turnout. The TopN algorithm, however, only generates one final path at the current hierarchy. At each time step, only the segment with the maximum gain in the next move grows (deducting a time step from the remaining flight time), until the remaining flight time is just enough to connect the two path segments. And with only one path generated, there is no need to search further at the current hierarchy.

Although simple, the Top2 and TopN algorithms become powerful when combined with the MGR heuristic and a hierarchical structure.

V. Experiment Results and Analysis

A. Experiment Setup

We selected three real WiSAR scenarios to test the performance of the proposed algorithms for ecological validity. All three scenarios were obtained from George Mason University, and all came from the International Search and Rescue Incident Database [28]. In each scenario, the missing person's last known position (LKP) is at the center of a 2.4×2.4 km search area; therefore, we always start the UAV path from the center of the map. The probability distribution map of the missing person for each scenario is generated using the Bayesian model presented in [30]. These probability distribution maps have been evaluated at George Mason University's MapScore web portal [31] and performed better than most other models evaluated.[3] The task difficulty map for each scenario is built using vegetation density data downloaded from the USGS website and categorized into three difficulty levels (sparse, medium, and dense). Although this method only considers the vegetation density, it gives us a reasonable task difficulty map and serves well for the purpose of demonstrating algorithm performances.[4] The probability distribution maps and the task difficulty maps are discretized into 100×100 grids.

For each scenario, we compare the performance of the BA, LHC-GW-CONV, Top2, and TopN algorithms in $\text{Efficiency}_{\text{LB}}$ and running time for three flight durations ($T = 300, 600, 900$, equivalent to 10, 20, and 30 min). Because we reimplemented the BA algorithm in MATLAB and the rest algorithms in C#, for a fair comparison, we omit the running time for the BA algorithm. We also present the performance of the Top2 and TopN algorithms for just one hierarchy layer to demonstrate that the two algorithms can achieve much better $\text{Efficiency}_{\text{LB}}$ in comparable running time with even arbitrary parameters ($k = 5$ Gaussians and $N = 3$ for top 3 subregions). In all the experiments, we did not specify the ending location for the UAV because the BA algorithm (the only one) does not support this feature.

Experiments were performed in simulated searches and not on-board real UAVs. All paths generated in the experiments were for a hexacopter although the algorithms also work for fixed-wing UAVs. All experiments were run on a Intel 4-core i7-2600 PC with 16 GB of memory. For each scenario, we ran 10 experiments and recorded the mean and standard deviation of $\text{Efficiency}_{\text{LB}}$ and running time. Due to space limitations, only a subset of the experiment results are presented.

B. Experiments Results and Analysis

In the first WiSAR scenario (*HikerPaul*), an elderly couple was reported missing near the Grayson Highlands State Park in Virginia (Fig. 9 shows a satellite imagery of the search area for the scenario). In the second WiSAR scenario (*NewYork*53), a 46-year-old male camper was reported missing near Adirondack Park in upper state New York. In the third WiSAR scenario (*NewYork*108), two teenage female hikers were reported missing near West Chesterfield in Massachusetts. For each scenario, the LKP of the missing person is in the center of the search region. Figs. 10, 13, and 15 show the probability distribution map (left) and the task difficulty map (middle) for these scenarios. The right part of the figures show the resulting

[3]Scoring 0.8184, 0.9858, and 0.9892 on a [-1, 1] scale, where the higher the score the better. Available at: http://sarbayes.org/projects/

[4]In real WiSAR operations, these maps would be further improved by domain experts before they are used for path planning.

Fig. 9. Satellite imagery of the search area for the *HikerPaul* scenario (near the Grayson Highlands State Park in Virginia) showing the vegetation density. The LKP of the missing person is at the center of the image.

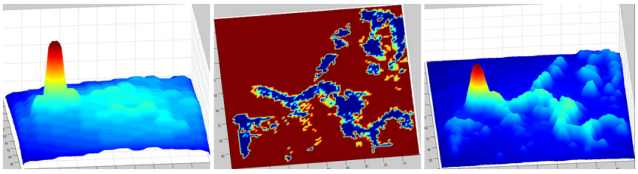

Fig. 10. *HikerPaul* scenario. (a) Probability distribution map. (b) Task difficulty map. (c) Surface after combining the probability distribution map and the task difficulty map.

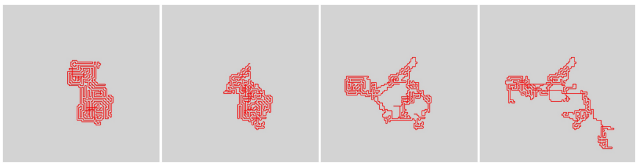

Fig. 11. Paths generated for *HikerPaul* scenario with T = 900. (a) BA. (b) LHC-GW-CONV. (c) Top2. (d) TopN.

surface for each scenario when we combine the probability distribution map and the task difficulty map, which is the amount of probability the UAV can collect on its first visit to each vertex (or grid cell). The task difficulty maps indicate that large areas of these search regions were covered with dense vegetation, which makes detecting the missing person more difficult. There are also small subregions with sparse vegetation (higher probability of detection). Figs. 11, 14, and 16 show the paths generated by the BA, LHC-GW-CONV, Top2, and TopN algorithms for each scenario, respectively. The teleport paths for these scenario are not shown because they are mostly made up of disjointed points. Note that the paths sometimes revisit vertices that have already been visited (path segments cross with previous segments), but the combined surfaces we show in Figs. 10, 13, and 15 (right) only represent the amount of probability the UAV can collect on its first visit. Each surface is updated after each vertex visit to reflect the amount of probability collectable on the next visit.

For the *HikerPaul* scenario, Fig. 11 shows that both the BA and the LHC-GW-CONV greedy-type algorithms generated paths that centered around the starting point and could not break away from the probability hills near the center of the search area. The Top2 algorithm, on the other hand, directed

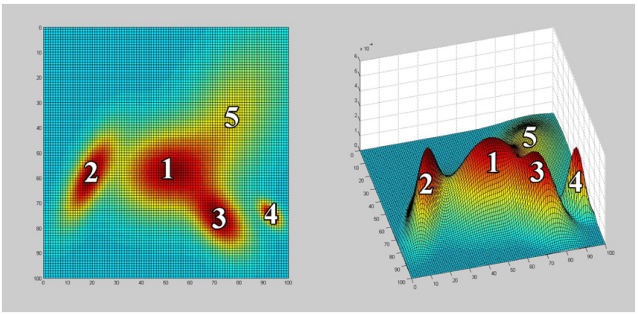

Fig. 12. Gaussian mixture identified for the *HikerPaul* scenario with T = 900 and k = 5. The numbers show the ranking of the Gaussians using MGR. (a) Gaussians in 2-D. (b) Gaussians in 3-D.

TABLE I

ALGORITHMS $\text{Efficiency}_{\text{LB}}$ AND RUNNING SPEED COMPARISON FOR THE *HikerPaul* SCENARIO (ALL NUMBERS SHOWN ARE AVERAGES OF 10 RUNS; ALL $\text{Efficiency}_{\text{LB}}$ STANDARD DEVIATIONS ARE BELOW 0.1)

	$Efficiency_{LB}$ (%)			Speed (seconds)		
T	300	600	900	300	600	900
BA	56.95	60.07	57.11	-	-	-
LHC-GW-CONV	60.18	56.76	55.18	0.30	0.47	0.98
Top2 (1 layer)	66.68	65.21	66.08	**0.24**	0.30	0.41
TopN (1 layer)	76.19	71.02	68.26	0.25	**0.24**	**0.22**
Top2 (Hierarchy)	78.67	73.81	72.75	0.73	0.84	1.19
TopN (Hierarchy)	**81.43**	**75.48**	**74.13**	1.52	1.73	1.68

the UAV to cover the tall probability hill on the left side of the search area, and the TopN algorithm additionally directed the UAV to cover subregions in the lower right of the search area where more probability can be accumulated. Fig. 12 demonstrates how a GMM can be used to prioritize search subregions and shows the five Gaussians identified when we performed the Gaussian fitting for the *HikerPaul* scenario with T = 900 and k = 5. The Gaussians are ranked using the MGR heuristic values (1.39, 1.01, 1, 0.87, and 0.46 respectively). Table I shows the performance of the four algorithms and also the Top2 and TopN algorithms with specific parameters (number of Gaussians to fit: k = 5 and top N subregions for TopN algorithm: N = 3). The Top2 and TopN algorithms clearly outperform the BA and LHC-GW-CONV algorithms (whether using arbitrary parameters or search through the hierarchy) with significantly better $\text{Efficiency}_{\text{LB}}$. Searching through the hierarchy generated more efficient paths than only working with one layer of the hierarchy. The TopN algorithm also achieved slightly better $\text{Efficiency}_{\text{LB}}$ than the Top2 algorithm. When using arbitrary parameters (only generating a path for one layer of the hierarchy), both the Top2 and TopN algorithms are faster than the LHC-GW-CONV algorithm. When searching through the hierarchy, the Top2 and TopN algorithm did take a little bit longer, but still completed within 2 s.

For the *NewYork*53 scenario, Fig. 14 shows that both the BA and LHC-GW-CONV greedy-type algorithms generated paths that spent a good amount of time right at the center of the search area around the starting point before sending the UAV to two other subregions on the right. The Top2 and TopN algorithms, by contrast, did not waste any time at the

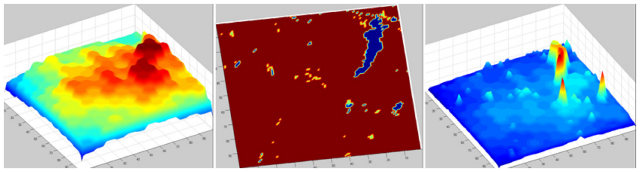

Fig. 13. *NewYork*53 scenario. (a) Probability distribution map. (b) Task difficulty map. (c) Surface after combining the probability distribution map and the task difficulty map.

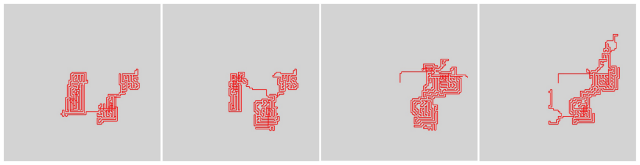

Fig. 14. Paths generated for *NewYork*53 scenario with T = 900. (a) BA. (b) LHC-GW-CONV. (c) Top2. (d) TopN.

TABLE II

ALGORITHMS $\text{Efficiency}_{\text{LB}}$ AND RUNNING SPEED COMPARISON FOR THE *NewYork*53 SCENARIO (ALL NUMBERS SHOWN ARE AVERAGES OF 10 RUNS; ALL $\text{Efficiency}_{\text{LB}}$ STANDARD DEVIATIONS ARE BELOW 0.07)

	$Efficiency_{LB}$ (%)			Speed (seconds)		
T	300	600	900	300	600	900
BA	39.95	54.27	65.08	-	-	-
LHC-GW-CONV	38.47	56.91	67.38	**0.01**	**0.02**	**0.02**
Top2 (1 layer)	54.42	66.61	72.79	0.75	0.92	0.81
TopN (1 layer)	59.15	68.78	74.54	0.70	0.77	0.69
Top2 (Hierarchy)	57.18	69.29	74.44	1.87	2.06	1.92
TopN (Hierarchy)	**65.39**	**71.47**	**77.36**	5.01	5.76	5.32

Fig. 15. *NewYork*108 scenario. (a) Probability distribution map. (b) Task difficulty map. (c) Surface after combining the probability distribution map and the task difficulty map.

Fig. 16. Paths generated for *NewYork*108 scenario with T = 900. (a) BA. (b) LHC-GW-CONV. (c) Top2. (d) TopN.

TABLE III

ALGORITHMS $\text{Efficiency}_{\text{LB}}$ AND RUNNING SPEED COMPARISON FOR THE *NewYork*108 SCENARIO (ALL NUMBERS SHOWN ARE AVERAGES OF 10 RUNS; ALL $\text{Efficiency}_{\text{LB}}$ STANDARD DEVIATIONS ARE BELOW 0.07)

	$Efficiency_{LB}$ (%)			Speed (seconds)		
T	300	600	900	300	600	900
BA	39.92	45.34	49.39	-	-	-
LHC-GW-CONV	41.38	52.88	52.61	**0.01**	**0.01**	**0.02**
Top2 (1 layer)	58.37	54.18	57.33	0.98	0.90	1.44
TopN (1 layer)	54.03	53.91	57.91	0.92	0.83	0.97
Top2 (Hierarchy)	**60.73**	55.91	57.94	2.42	2.52	2.50
TopN (Hierarchy)	59.60	**60.26**	**60.99**	6.81	6.59	7.42

center subregion and immediately directed the UAV to cover the two subregions on the right side of the search area. The TopN algorithm also directed the UAV to cover a subregion in the upper right part of the search area. Table II shows the performance of the four algorithms and also the Top2 and TopN algorithms with specific parameters (k = 5 and N = 3). The results show the same trend as with the first scenario where the Top2 and TopN algorithms outperform the BA and LHC-GW-CONV algorithms significantly in $\text{Efficiency}_{\text{LB}}$. Even with arbitrary parameters, the Top2 and TopN algorithms generated much more efficient paths (e.g., 59.15% for TopN with one layer versus BA with 39.95%). The TopN algorithm also outperformed Top2 algorithm in $\text{Efficiency}_{\text{LB}}$. When looking at the algorithm completion time, LHC-GW-CONV algorithm is the clear winner in this scenario. When arbitrary parameters are used, the Top2 and TopN algorithms both completed within 1 s, but when searching through the hierarchy, both algorithms took much longer (about 2 s for Top2 and 6 s for TopN) to complete.

For the *NewYork*108 scenario, Fig. 16 shows that both the BA and LHC-GW-CONV greedy-type algorithms generated paths that spent a good amount of time at the center of the search area around the starting point before moving on to the upper right subregion of the search area to cover the probability ridge. The Top2 and TopN algorithms, however, did not waste any time at the center subregion and immediately directed the UAV to cover the probability ridge at the upper right subregion of the search area. Both of them also sent the UAV to another subregion at the lower left part of the search area where a good amount of probability can be collected. Table III shows the performance of the four algorithms and also the Top2 and TopN algorithms with specific parameters (k = 5 and N = 3). The results show the same trend as with the previous two scenarios where the Top2 and TopN algorithms outperform the BA and LHC-GW-CONV algorithms significantly in $\text{Efficiency}_{\text{LB}}$. Even with arbitrary parameters, the Top2 and TopN algorithms generated more efficient paths (e.g., 58.37% for Top2 with one layer versus BA with 39.92%). In this scenario, the Top2 algorithm performed slightly better than the TopN algorithm in $\text{Efficiency}_{\text{LB}}$ at T = 300 (60.73% for Top2 and 59.60% for TopN), but the TopN algorithm performed much better than the Top2 algorithm for the other two cases. When looking at the algorithm completion time, LHC-GW-CONV algorithm is still the clear winner in this scenario. When arbitrary parameters are used, the Top2 and TopN algorithms both completed in about 1 s, but when searching through the hierarchy, both algorithms took much longer (about 2.5 s for Top2 and 7 s for TopN) to complete.

Note that the performance metric $\text{Efficiency}_{\text{LB}}$ is computed using (9), which assumes that the UAV can teleport within the search area. Because the amount of probability accumulated following this teleporting path can be much better than the optimal path, the true search efficiency is likely much better than the value of the $\text{Efficiency}_{\text{LB}}$. Fig. 17 shows the comparison

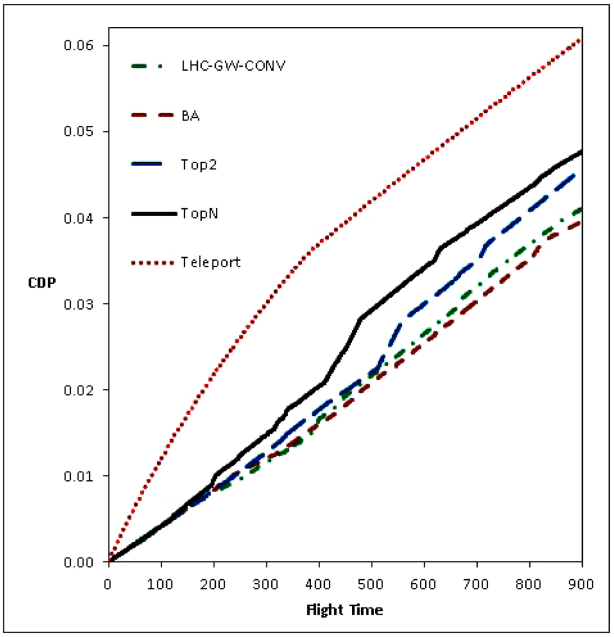

Fig. 17. Paths CDPs comparison at T=900 with partial detection.

of the algorithms performance with respect to CDP collected over time for the *NewYork*53 scenario when $T = 900$. The dotted red line represents CDP accumulated over time if the UAV could teleport from vertex to vertex. Therefore, this line represents the theoretical CDP upperbound. If we know the optimal path and can plot the performance, that line would most likely be somewhere below the teleport path line. The TopN algorithm (black solid line) performed the best (highest CDP value at time step 900), and the Top2 algorithm (blue dash line) ranked second. Both the Top2 and TopN algorithms outperformed the BA and LHC-GW-CONV algorithms (the bottom two dashed lines) significantly.

Across experiments, results show that by taking advantage of the MGR heuristic, the Top2 and TopN algorithms (even when arbitrary parameters, $k = 5$ Gaussians to fit and $N = 3$ for top 3 subregions, are used) always generated more efficient paths than those generated by the BA and LHC-GW-CONV algorithms. When hierarchical search is performed, the improvement from Top2 and TopN algorithms is significant. In most cases, the TopN algorithm outperformed the Top2 algorithm. However, when hierarchical search is performed, the Top2 and TopN algorithms did take a little longer to complete.

VI. Limitations and Discussion

In our problem formulation, we treat the missing person as stationary because the speed of the missing person in wilderness is relatively low when compared with the speed the UAV travels in. False detection is not an issue because the UAV can simply follow the path generated to continuously collect detection probability while human operators verify the accuracy of the detection. For other application domains where the target might be moving or the probability distribution might be changing during search, by setting T to a small value, we can easily adapt the Top2 and TopN algorithms to handle these situations. The two algorithms effectively turn into greedy (a T-step look ahead approach compared to the one-step look ahead method in [7]) algorithms with flexible time horizons and scalability. We leave the evaluation of the two algorithms in such scenarios to future work.

In (4), we assume that each observation at vertex v_i is conditionally independent. This assumption certainly has its limitation. If the environment features remain the same (e.g., lighting conditions and vegetation density) and the sensor platform (e.g., camera) has stable performance, then a high probability of no detection on the first visit might indicate high probability of no detection on future visits. However, in practical applications, a sensor operator's ability to recognize the missing person from video footprint is affected by many factors such as his fatigue level and cognitive workload [1], especially when the sensor operator might also be flying the UAV. In this case, the operator's chance performance can be regarded as independent trials (as in successive coin tosses).

The detection model used in our experiments is a simple decay model only parameterized by a difficulty factor. In SAR literature, the parameters of the decay factor could be affected by environment features and sensor properties (e.g., distance to radar and signal strength [7]). Also we only consider vegetation density when we constructed the task difficulty maps. Because we use a camera sensor and keep the UAV flying at the same height above ground, we believe this model is sufficient to show the algorithms' capability in handling partial detection, and we intentionally kept the sensor model and environment model simple for demonstration purposes. However, a more complicated sensor model and environment model can easily be applied to the system.

Although GMM is a statistically mature method for clustering, it has several limitations. First, convergence is not guaranteed for the iterative EM algorithm used to estimate the Gaussian mixture. In our implementation, we re-run GMM multiple times if convergence is not achieved to overcome the problem. Second, how many Gaussians should we fit? There is the possibility that the Gaussians might not fit the data very well. We arbitrarily set the maximum Gaussians to 5 to reduce computational complexity. Experiment results show that we were still able to generate good paths. Since the MGR is only a heuristic, as long as it provides useful information to our search most of the time, it serves its purpose.

Another limitation is that the algorithms do not handle tough terrains where the UAV might not be able to climb fast enough to fly over the terrain. Future work should explore how to modify the algorithms to consider such constraints and actual flight dynamics.

When defining the goodness of a subregion, MG_i, we considered three factors: distance ratio, probability volume, and subregion area. The last two factors, when combined, give us a sense of the peakedness of a probability hill. Then we multiply the peakedness with the distance ratio in order to compare the MG of subregions. Here, we assume the two measures are independent of each other, which is a limitation of the heuristic. It also creates a trade-off problem. For example, when the distance ratio of the subregion A is half of that

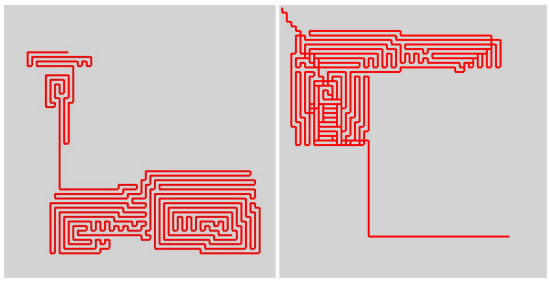

Fig. 18. Paths found for the synthetic scenario at T = 900 when the UAV starts from a subregion with high task difficulty (lower right corner). (a) Path created by LHC-GW-CONV without specifying end point. (b) Path created by LHC-GW-CONV with specified end point at upper left corner.

TABLE IV
ALGORITHMS $Efficiency_{LB}$ COMPARISON FOR THE MULTIMODAL SYNTHETIC SCENARIO AT T = 900

(%)	BA	LHC-GW-CONV	Top2	TopN
Start from upper left	65.99	**96.28**	94.96	95.82
Start from lower right	41.91	53.71	93.46	**95.29**

of subregion B but peakedness of A is twice in size compared to B, A and B would still have identical MG_i values. Should they be? We leave this to future work.

Going back to the synthetic scenario, we presented in Section IV-A, Table IV shows the performance of the BA, LHC-GW-CONV, Top2, and TopN algorithms in two different scenarios: starting from a subregion with low task difficulty (upper left) and starting from a subregion with high task difficulty (lower right). When starting from a high task difficulty area, the BA and LHC-GW-CONV algorithms tend to get stuck in a local probability hill, while the Top2 and TopN algorithms force the UAV to visit other subregions, therefore achieving better paths with significant improvement. One interesting observation we noticed is that if an ending position is specified for the desired UAV path and the ending point is in a subregion with low task difficulty, the LHC-GW-CONV algorithm also forces the UAV to visit other subregions, and by doing so, improves the efficiency of the path. Fig. 18 shows an example where the path on the right achieved 93.60% in $\text{Efficiency}_{\text{LB}}$ (computed within 0.01 s), which is slightly better than the Top2 algorithm but not as good as the TopN algorithm. Another thing to note with this scenario is that the LHC-GW-CONV algorithm actually did slightly better than the Top2 and TopN algorithms when the UAV starts from a low task difficulty area. We have noticed from various experiments that after combining the probability distribution map and the task difficulty map, if the resulting surface is not a complicated one (meaning it only has a few distinctive probability hills), the LHC-GW-CONV algorithm generally performs well. For more complicated surfaces (such as the three real WiSAR scenarios we tested the algorithms with), the Top2 and TopN algorithms are more reliable in generating good UAV paths.

In the current implementation, we used a grid representation for the probability distribution map, task difficulty map, and the path generated. However, the algorithms also support other tessellation methods such as a hexagonal tessellation.

VII. CONCLUSION

We proposed a new heuristic, the MGR, which uses GMM to prioritize search subregions, and presented two new algorithms that utilize the heuristic in hierarchical path planning. The hierarchical structure enables searching for better paths through the parameter space at different scales and enables us to parallelize the two algorithms for better performance. The probability of detecting the desired subject based on UAV sensor information can vary in different search areas due to factors such as varying vegetation density or lighting conditions. We represented this type of partial detection using a task difficulty map, a spatial representation of sensor detection probability, and incorporated it into UAV path planning. We compared the performance of the new algorithms against two published algorithms, BA and LHC-GW-CONV, in simulated searches with three real SAR scenarios. Experiment results showed that by using the MGR heuristic, the two new algorithms Top2 and TopN consistently outperform the BA and LHC-GW-CONV algorithms, yielding efficient paths that produce payoff approximating the payoff of the optimal path.

REFERENCES

[1] M. A. Goodrich *et al.*, "Supporting wilderness search and rescue using a camera-equipped mini UAV," *J. Field Robot.*, vol. 25, no. 1–2, pp. 89–110, Jan. 2008.

[2] W. G. Syrotuck, *An Introduction to Land Search Probabilities and Calculations*. Mechanicsburg, PA, USA: Barkleigh Productions, 2000.

[3] B. O. Koopman, "The theory of search. III: The optimum distribution of searching effort," *Oper. Res.*, vol. 5, pp. 613–626, Oct. 1957.

[4] L. D. Stone, *Theory of Optimal Search*. New York, NY, USA: Academic, 1975.

[5] A. R. Washburn, *Search and Detection*, Military Applications Section, Operations Research Society of America, Arlington, VA, USA, 1981.

[6] B. S. Morse, C. H. Engh, and M. A. Goodrich, "UAV video coverage quality maps and prioritized indexing for wilderness search and rescue," in *Proc. 5th ACM/IEEE Int. Conf. Human-Robot Interact.*, Mar. 2010, pp. 227–234.

[7] F. Bourgault, T. Furukawa, and H. F. Durrant-Whyte, *Optimal Search for a Lost Target in a Bayesian World*, ser. Springer Tracts in Advanced Robotics, vol. 24. New York, NY, USA: Springer-Verlag, 2006, ch. 21, pp. 209–222.

[8] L. Lin and M. A. Goodrich, "UAV intelligent path planning for wilderness search and rescue," in *Proc. IROS*, Oct. 2009, pp. 709–714.

[9] K. E. Trummel and J. R. Weisinger, "Technical note: The complexity of the optimal searcher path problem," *Oper. Res.*, vol. 34, no. 2, pp. 324–327, Mar. 1986.

[10] M. Quigley, B. Barber, S. Griffiths, and M. A. Goodrich, "Towards real-world searching with fixed-wing mini-UAVs," in *Proc. IROS*, Aug. 2005, pp. 3028–3033.

[11] A. Stentz, *Optimal and Efficient Path Planning for Partially Known Environments*. New York, NY, USA: Springer, 1997, ch. 11, pp. 203–220.

[12] S. A. Bortoff, "Path planning for UAVs," in *Proc. Am. Contr. Conf.*, vol. 1, no. 6. Sep. 2000, pp. 364–368.

[13] P. O. Pettersson and P. Doherty, "Probabilistic roadmap based path planning for an autonomous unmanned helicopter," *J. Intell. Fuzzy Syst.*, vol. 17, no. 4, pp. 395–405, 2006.

[14] R. C. Holte, M. B. Perez, R. M. Zimmer, and A. J. MacDonald, "Hierarchical A*: Searching abstraction hierarchies efficiently," in *Proc. AAAI/IAAI*, vol. 1. 1996, pp. 530–535.

[15] N. Meuleau and R. I. Brafman, "Hierarchical heuristic forward search in stochastic domains," in *Proc. IJCAI*, vol. 7. 2007, pp. 2542–2549.

[16] M. Naveed, D. E. Kitchin, and A. Crampton, "A hierarchical task network planner for pathfinding in real-time strategy games," in *Proc. 3rd Int. Symp. AI & Games*, 2010, pp. 1–7.

[17] G. Laporte, "The vehicle routing problem: An overview of the exact and approximate algorithms," *Eur. J. Oper. Res.*, vol. 59, no. 3, pp. 345–358, Jun. 1992.

[18] P. R. Sokkappa, "The cost-constrained traveling salesman problem," Ph.D. Dissertation, Univ. California, Berkeley, CA, USA, Oct. 1990.
[19] M. F. Tasgetiren and A. E. Smith, "A genetic algorithm for the orienteering problem," in *Proc. Congr. Evol. Computat.*, vol. 2. 2000, pp. 910–915.
[20] Y.-C. Liang and A. E. Smith, "An ant colony approach to the orienteering problem," *J. Chinese Inst. Industr. Eng.*, vol. 23, no. 5, pp. 403–414, Jan. 2006.
[21] M. Sniedovich, *Dynamic Programming: Foundations and Principles*. Boca Raton, FL, USA: CRC Press, 2010.
[22] D. P. Bertsekas, *Dynamic Programming and Optimal Control*, vol. 1, no. 2. Belmont, CA, USA: Athena Scientific, 1995.
[23] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, vol. 1, no. 1. Cambridge, U.K.: Cambridge Univ. Press, 1998.
[24] B. O. Koopman, "The theory of search. II. Target detection," *Oper. Res.*, vol. 4, no. 5, pp. 503–531, Oct. 1956.
[25] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*, vol. 1. Cambridge, MA, USA: MIT Press, 2005.
[26] P. Niedfeldt, R. Beard, B. S. Morse, and S. Pledgie, "Integrated sensor guidance using probability of object identification," in *Proc. ACC*, Jun./Jul. 2010, pp. 788–793.
[27] A. Ryan and J. K. Hedrick, "Particle filter based information-theoretic active sensing," *Robot. Auton. Syst.*, vol. 58, no. 5, pp. 574–584, May 2010.
[28] R. J. Koester, *Lost Person Behavior: A Search and Rescue*. Charlottesville, VA, USA: dbS Productions LLC, 2008.
[29] D. Ferguson, "GIS for wilderness search and rescue," in *Proc. ESRI Federal User Conf.*, Feb. 2008.
[30] L. Lin and M. A. Goodrich, "A Bayesian approach to modeling lost person behaviors based on terrain features in wilderness search and rescue," *Computat. Math. Organiz. Theory*, vol. 16, no. 3, pp. 300–323, Sep. 2010.
[31] E. Cawi, N. Jones, and C. R. Twardy, "Mapscore: Probability map evaluation for search & rescue," *Proc. Conf. Presentat. Virginia Search Rescue*, Holiday Lake, VA, USA, 2012.
[32] L. D. Stone, C. Keller, T. M. Kratzke, and J. P. Strumpfer, "Search analysis for the location of the AF447 underwater wreckage," *Report to BEA*, pp. 1–8, 2011.
[33] K. P. Balanda and H. L. MacGillivray, "Kurtosis: A critical review," *Am. Statist.*, vol. 42, no. 2, pp. 111–119, 1988.
[34] K. Mardia, "Measures of multivariate skewness and kurtosis with applications," *Biometrika*, vol. 57, no. 3, p. 519, 1970.
[35] A. Khurshid, E. Hussain, and M. ul Haq, "A note on finding peakedness in bivariate normal distribution using mathematica," *Pak. J. Statist. Oper. Res.*, vol. 3, no. 2, pp. 75–86, 2007.
[36] P. S. Horn, "A measure for peakedness," *Am. Statist.*, vol. 37, no. 1, pp. 55–56, 1983.
[37] M. R. Gupta and Y. Chen, *Theory and Use of the EM Algorithm*. Hanover, MA, USA: Now Publishers Inc., 2011.

Lanny Lin (M'13) received the B.S. degree in computer science and the B.A. degree in business from Southern Oregon University, Ashland, OR, USA, in 1997, and the M.S. degree in computer science from Brigham Young University, Provo, UT, USA, in 2009. He is currently pursuing the Ph.D. degree in computer science at Brigham Young University.

His current research interests include human–robot interaction, Bayesian model design, algorithm design, and robotics.

Michael A. Goodrich (SM'14) received the B.S., M.S., and Ph.D. degrees in electrical and computer engineering from Brigham Young University, Provo, UT, USA, in 1992, 1995, and 1996, respectively.

From 1996 to 1998, he was a Research Associate with Nissan Cambridge Research, Nissan Research and Development, Inc., Cambridge, MA, USA. Since 1998, he has been with the Department of Computer Science, Brigham Young University, where he is currently a Professor. His current research interests include human–robot interaction, decision theory, multiagent learning, and human-centered engineering.