Geolocation of RF Emitters by Many UAVs

Paul Scerri, Robin Glinton, Sean Owens, David Scerri and Katia Sycara
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213, USA
{pscerri, rglinton, owens}@cs.cmu.edu, dscerri@gmail.com, katia@cs.cmu.edu

This paper presents an approach to using a large team of UAVs to find radio frequency (RF) emitting targets in a large area. Small, inexpensive UAVs that can collectively and rapidly determine the approximate location of intermittently broadcasting and mobile RF emitters have a range of applications in both military, e.g., for finding SAM batteries, and civilian, e.g., for finding lost hikers, domains. Received Signal Strength Indicator (RSSI) sensors on board the UAVs measure the strength of RF signals across a range of frequencies. The signals, although noisy and ambiguous due to structural noise, e.g., multi-path effects, overlapping signals and sensor noise, allow estimates to be made of emitter locations. Generating a probability distribution over emitter locations requires integrating multiple signals from different UAVs into a Bayesian filter, hence requiring cooperation between the UAVs. Once likely target locations are identified, EO-camera equipped UAVs must be tasked to provide a video stream of the area to allow a user to identify the emitter.

I. Introduction

The rapidly improving availability of small, unmanned aerial vehicles (UAVs) and their ever reducing cost is leading to considerable interest in multi-UAV applications. However, while UAVs have become smaller and cheaper, there is a lack of sensors that are light, small and power efficient enough to be used on a small UAV yet are capable of taking useful measurements of objects often several hundred metres below them. Static or video cameras are one option, however image processing normally requires human input or at least computationally intensive offboard processing, restricting their applicability to very small UAV teams. In this paper, we look at how teams of UAVs can use very small Received Signal Strength Indicator (RSSI) sensors whose only capability is to detect the approximate strength of a Radio Frequency (RF) signal, to search for and accurately locate such sources. RSSI sensors give at most an approximate range to an RF emitter and will be misleading when signals overlap. Applications of such sensors range from finding lost hikers or skiers carrying small RF beacons to military reconnaissance operations. Moreover, the basic techniques have a wider applicability to a range of robotic teams that rely on highly uncertain sensors, e.g., search and rescue in disaster environments.

Many of the key technogies required to build a UAV team for multi-UAV applications have been developed and are reasonably mature and effective.[1,2] However, for large UAV teams with very noisy sensors, key problems remain, specifically, much previous work is formally grounded but impractical.[3] Often the coordination and planning algorithms and the representations of the environment are not appropriate for more than two or three UAVs and targets. For example, some solutions require a UAV to know the planned paths of all other UAVs in order to plan its own path,[8] but this is infeasible (both in terms of communication and computation) when the number of UAVs is large. Other approaches only solve part of the problem, e.g., estimating locations from sensor readings[13,14] or planning cooperative paths,[11] but do not combine these elements in an integrated solution, although there are some exceptions.[4] Signal processing techniques for creating probability distributions from noisy signals have been extensively studied, but rarely have distributed filters versions been created and those that have been do not scale to larger teams.[9]

This paper presents an integrated approach to the problem that leverages three key ideas. The first key idea is to use a Binary Bayesian Grid Filter (BBGF)[7] to represent the probability that there is an emitter at any particular location. Each UAV maintains its own BBGF and anonymously forwards a select subset of sensor readings to some of the other UAVs.[12] The output of the BBGF is transformed into a map of *information entropy*. The UAVs plan paths through areas of highest entropy, thus when they fly those paths they can expect to maximize their information gain. The UAVs share their planned paths with team members, who take those paths into account when planning. Specifically, the UAV assumes that readings will be taken along that path and the expected change in entropy due to those readings is factored into its planning.

The second key idea is to use a Rapidly-expanding Random Tree (RRT) planner[6] to determine UAV paths. Such planners typically require a start and end location and a representation of the cost of traversing any terrain. In the information maximizing approach, there is no end location, rather the UAV should fly where sensor readings can be taken that will most improve the team's model of the environment. The terrain can also be taken into account, to encourage searching areas where emitters are more likely to be found first. The approach taken here is to represent each of the influences on the planned path, i.e., entropy, the paths of others and terrain, as a *cost map* which are combined and used by the RRT planner to determine the utility of many paths. Initial experiments show that the planner is effective and efficient at creating paths for the UAVs.

The third key idea is to use clustering algorithms on the BBGF to determine likely emitter locations. These locations are used to automatically cause a UAV equipped with an EO-camera to go to the area and provide a video stream for a user to locate the target. If the BBGF can reasonably accurately geo-locate the emitters, then the amount of video an operator needs to inspect can be dramatically reduced, thus increasing the total area the combined UAV-human system can search.

The overall approach described above has been implemented within the Machinetta[10] proxy infrastructure and tested in a medium fidelity simulation environment. The coordination reasoning and state estimation code is separated from the auto-pilot code so that it is applicable to a range of UAVs capable of waypoint following. Flight tests on real UAVs, in cooperation with an industrial partner validated the approach. Figure 1 shows the control interface, with the map display on the left, simulated camera view at top right and filter output at the bottom right. Simulation experiments show the approach to be effective at identifying likely emitter locations and sending EO-camera equipped UAVs to areas as small as 100m across.

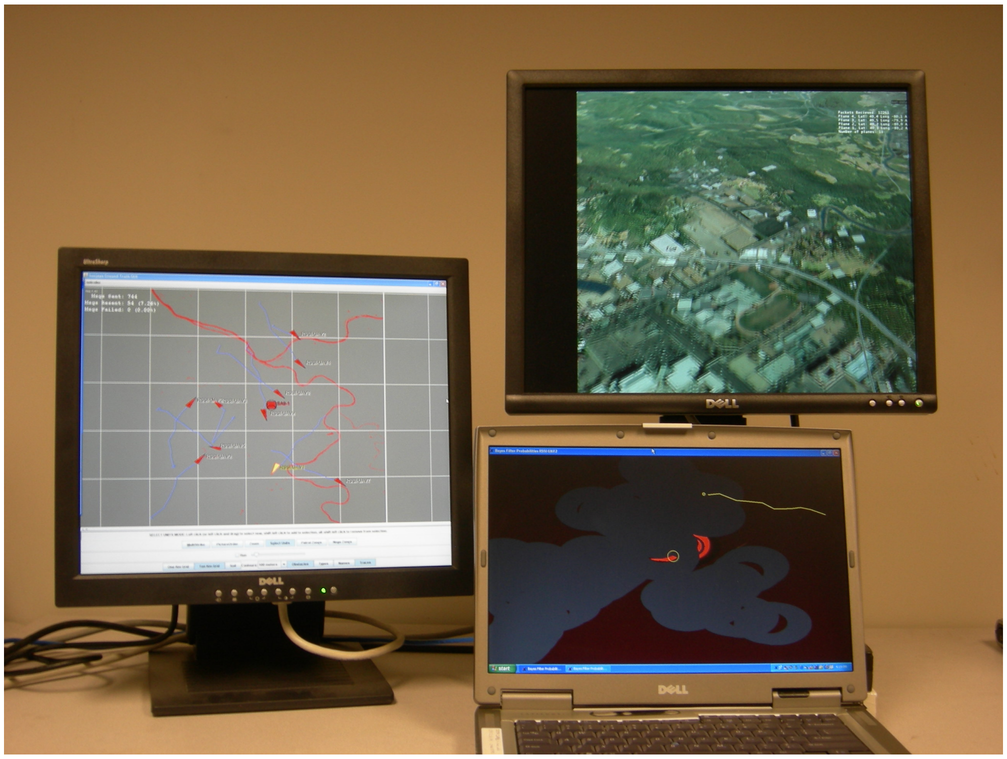

Figure 1. Operator interface station showing map view, camera view and filter output.

II. Problem

This article presents a method for localizing an unknown number of RF emitters using a team of UAVs. Most UAVs are outfitted wth RSSI sensors which detect the power of an RF signal at a position in space. A small number of UAVs are outfitted with EO sensors capable of streaming video back to a user. The environment is assumed to be too large to be feasibly search using EO sensors alone. The UAVs must maintain a belief over the state of all emitters in the environment in a decentralized manner.

The emitters are represented by the set: $E = \{e_1 \dots e_n\}$ where n is not known to the team of UAVs. Emitters are all assumed to be emitting at a single known frequency.[a] Emitters are mobile and emit intermittently. The homogeneous UAVs are represented by the set: $U = \{u_1 \dots u_m\}$. Each u_i flies a path given by $\vec{u}^i(t)$. During flight a UAV takes sensor readings, $z_t(\vec{loc})$ which are the received signal power at a location $\vec{loc} = \{x, y, z\}$ where $\{x, y, z\}$ gives the Euclidean coordinates of a point in space relative to a fixed origin. The power of the signal received is a result of three components. The first component, $\Gamma(\vec{loc}, e_i) = \frac{e_{const}}{dist(\vec{loc}, e_i)^2}$, where $dist(loc, e_i)$ is the Euclidean distance between loc and e_i and e_{const} is a constant that gives the power at $dist(loc_{e_1}, e_i) = 0$, is due to the sources themselves. The second component, $EN(\vec{loc}, E)$, is due to multi-path and attenuation of the signal due to environmental factors. Multi-path occurs when a reflected component of the signal arrives at a receiver and in combination with an attenuated direct signal results in a perturbed source location estimate. Finally ϵ gives typical zero-mean normally distributed sensor noise. The total power received at a location (loc) in space is then given by:

$$z_t(\vec{loc}) = \sum_{e_i \in E} \Gamma(\vec{loc}, e_i) + EN(\vec{loc}, E) + \epsilon \sim \mathcal{N}(0, \sigma)$$

Figure 2 shows some signals that will be received at different distances from a single emitter (i.e., no overlap). This is the basic signal model used in the simulation results below and closely represents real data collected from RSSI sensors on a physical UAV. The x-axis shows the distance and the y-axis shows the signal strength in Db (which is a log scale.) There are two important things to notice about this signal. First, it is very noisy, with high variation at all distances from the emitter, with some background noise high enough to represent being close to the emitter. Second, it has a very long "tail", i.e., at a reasonable distance from the emitter there is still useful information in the signal. Figure 3 shows the sensor readings when the UAV flies near one emitter and then another. Notice the overlap in the signals between the emitters, which are about 350m apart.

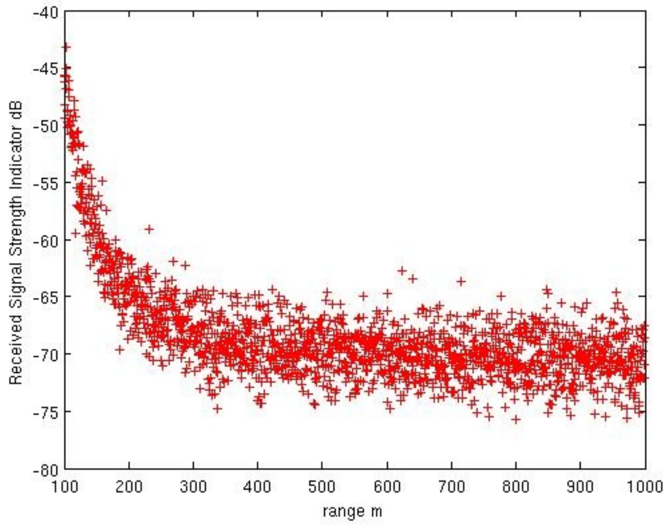

Figure 2. Sensor readings taken from different distances from an RF emitter.

The sensor readings taken by the ith UAV, up until time t are $z^i_{t_0} \dots z^i_t$. Each UAV maintains a posterior distribution P over emitter locations given by $P^i_t(e_1 \dots e_n | z^i_{t_0} \dots z^i_t)$. The UAVs proactively share sensor

[a]This will be relaxed in future work.

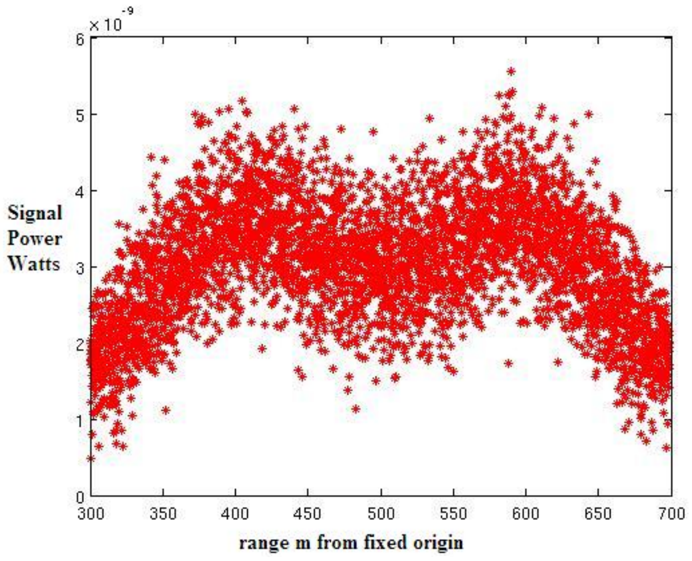

Figure 3. Sensor readings taken when flying between two emitters, first near one, then near the other.

readings to improve each other's posterior distribution. At time t each u_i can send some subset of locally sensed readings: $\vec{z}_t^i \subset z_{t_0}^i \dots z_t^i$.

The true configuration of the emitters in the environment at time t is represented as a distribution Q such that

$$Q_t(e_1 \dots e_n) = 1$$

when $e_1 \dots e_n$ gives the true configuration of the emitters at t. The objective is to minimize the divergence between the team belief and the true state of the emitters, while minimizing the cost of UAV flight path, and minimizing the total number of messages shared between UAVs. The following function expresses this mathematically:

$$\min_{\vec{u}^i} \sum_t \sum_{u_i \in U} \beta_1 Cost(\vec{u}^i(t)) + \beta_2 D_{\mathrm{KL}}(P_t^i \| Q) + \beta_3 |\vec{z}_t^i|$$

where D_{KL} denotes the the Kullback Leibler divergence and $\beta_{1\dots3}$ are weights which control the importance of the individual factors in the optimization process.

When P indicates the location of emitters to within a "reasonable" distance, EO assets should be sent to the estimated location to provide a video feed back to an operator who can determine the nature of the emitter. The earlier the EO assets are taking video of the emitter the better. However, the advantage of sending EO assets early can be mitigated by requiring that too large of an area be covered by video or by sending assets to areas where there are no emitters, i.e., false positives.

III. Algorithm

The most important feature of the overall algorithm is the tight integration of all the key elements to maximize performance at a reasonable computational and communication cost. A Binary, Bayesian Grid Filter (BBGF) maintains an estimate of the current locations of any RF emitters in the environment. This distribution is translated into a map of the entropy in the environment. The entropy is captured in a *cost map*. UAVs plan paths with a modified Rapidly-expanding Randomized Tree (RRT) planner that maximize the expected change in entropy that will occur due to flying a particular path. The most important incoming sensor readings, as computed by the KL information gain they cause, are forwarded to other members of the team for integration into the BBGFs of other UAVs. Planned paths are also shared so that other UAVs can take into account the expected entropy gain of other UAVs when planning their own paths. The paths of other UAVs are also captured in a *cost map*. Additional cost maps, perhaps capturing results of terrain analysis or no-fly zones, can be easily added to the planner.

The hardware independent components (planners, filters, etc.) are isolated from the hardware specific components (sensor drivers, autopilot) to allow the approach to be quickly integrated with different UAVs or moved from simulation to physical UAVs. The hardware independent components are encapsulated in a *proxy* which will either be on the physical UAV or on a UAV ground station, depending on the vehicle. In the experiments below, *exactly* the same proxy code is used in simulation as will be used in tests with physical UAVs. Figure 4 shows the main components and information flows from the perspective of one UAV-proxy.

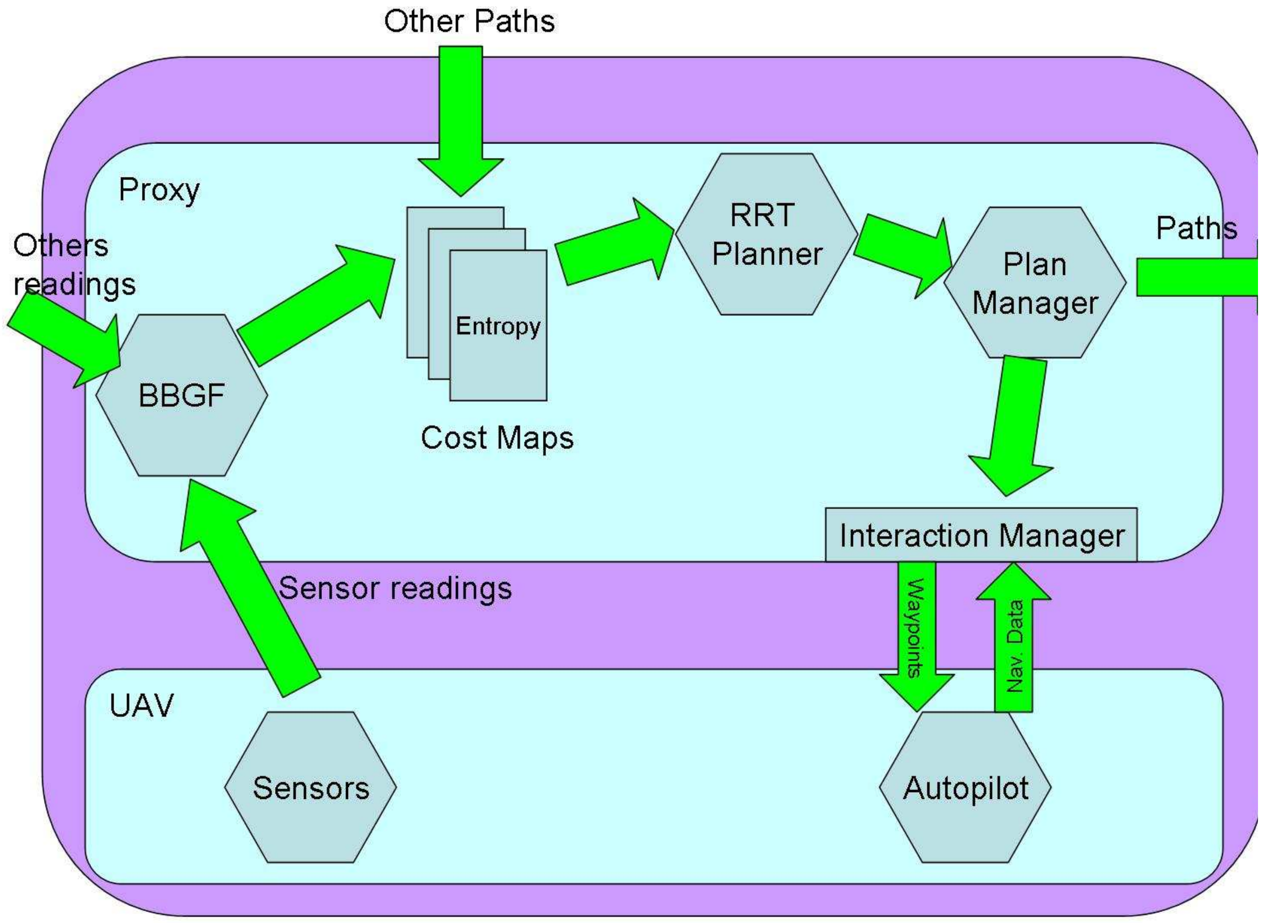

Figure 4. Block diagram of architecture.

IV. Distributed State Estimation

In this section, we describe the filter used to estimate the locations of the emitters. In previous work we have described the algorithms that UAVs use to decide which sensor readings to share with other UAVs, to ensure that each member of the team has the best possible estimate of emitter locations while respecting bandwidth limitations.[12]

IV.A. Binary, Bayesian Grid Filter

The filter uses a grid representation, where each cell in the grid represents the probability that there is an emitter in the area on the ground corresponding to that location.[b] For a grid cell c the probability that it

[b]A quad-tree or other representation might reduce memory and computational requirements in very large environments, but the algorithmic complexity is not justified for reasonable sized domains.

contains an emitter is written $P(c)$. The grid as a whole acts as the posterior $P_t^i(e_1 \dots e_n | z_{t_0}^i \dots z_t^i)$.

To make calculations efficient, we represent probabilities in *log odds* form, i.e., $l_t = logP(i)$. Updates on grid cells are done in a straightfoward Bayesian manner.

$$l_t = l_{t-1} + log\frac{P(e_i|z_t)}{1 - P(e_i|z_t)} - log\frac{P(e_i)}{1 - P(e_i)}$$

where $P(e_i|z_t)$ is a inversion of the the signal model, with the standard deviation extended for higher powered signals, i.e.,

$$P(e_i|z_t) = \begin{cases} \frac{1}{\sqrt{2\pi(\sigma_1^2)}} e^{-\frac{1}{2}(z_t - \Gamma)^2} & \text{if } z_t \geq \Gamma \\ \frac{1}{\sqrt{2\pi(\sigma_2^2)}} e^{-\frac{1}{2}(z_t - \Gamma)^2} & \text{otherwise} \end{cases}$$

where $\sigma_1 > \sigma_2$ scales the standard deviation on the noise to take into account structural environmental noise and overlapping signals. Intuitively, overlapping and other effects might make the signal stronger than expected, but they are less likely to make the signal weaker than expected. Figure 5 shows a plot of the (log) probability (y-axis) of a signal of a particular strength (x-axis) when the emitter is 500 m from the sensor.

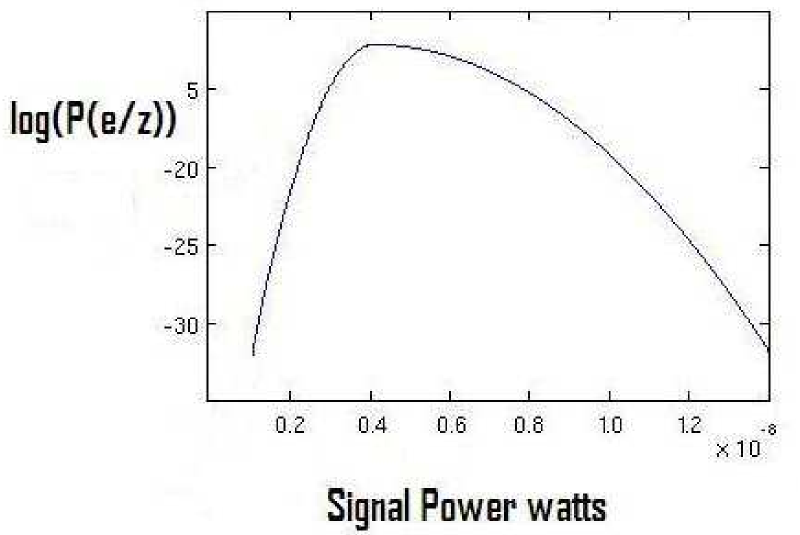

Figure 5. Mapping between probability and signal strength.

Notice that there is no normalization process across the grid because the number of emitters is not known. If the number of emitters were known, a normalization process might be able to change the probability of emitters even in areas where no sensor readings had been taken. Initial values of grid cells are set to values reflecting any prior knowledge or some small uniform value if no knowledge is available.

The UAVs will fly to areas of maximum entropy, hence the probability distribution has to be translated into an entropy distribution. We assume independence between grid cells, so entropy can be calculated on a grid cell by grid cell basis. Specifically, the entropy, H, of a grid cell i is:

$$H(i) = P(i)log(P(i)) + (1 - P(i))log(1 - P(i))$$

Figure 6 shows how probability and entropy are related.

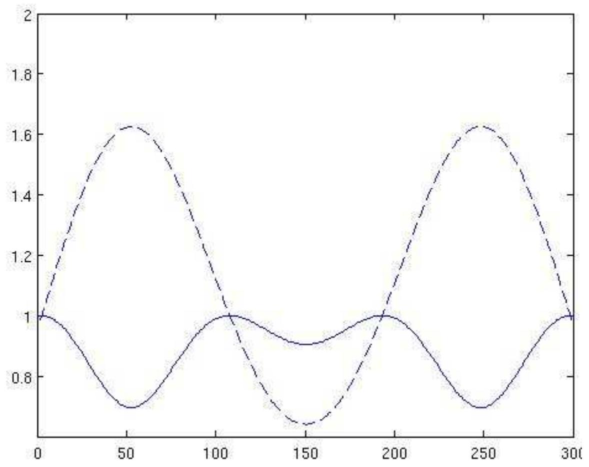

Figure 6. Mapping between probability of an emitter and entropy. The broken line shows the probability and the unbroken line the entropy.

V. Identifying Targets in the Bayesian Grid

It is necessary to go from the BBGF and its grid of probabilities to a representation that can be used for tasking of assets to image an emitter. This requires that a decision is made, based on the probability distribution, about where emitters are most likely to be found. The following is a description of the process used to accomplish the necessary transformation between representations.

Recall that the BBGF represents the posterior $P_t^i(e_1 \dots e_n | z_{t_0}^i \dots z_t^i)$ over emitter location by the set $P = \{P(c_1) \dots P(c_n)\}$ where c_1 is a cell in an mxn grid G and G is a planar discretization of the environment being searched. $P(c_1)$ gives the probability that a geographical location that corresponds to grid cell $c_1 \in G$ contains an emitter.

The first step is to threshold the BBGF. The result is a set of cells $P_{high} \subset P$, where $P_{high} = \{c_i \in G : P(c_i) > b\}$ and b is a user selected threshold on the probability that an emitter is present.

Next, the algorithm AllocateSensor takes as input the set P_{high} and produces as output a list $EOList$ which contains a set of error ellipses, the means of which are locations that EO sensors are tasked to image. The algorithim proceeds by clustering the elements of P_{high}. In this work the *Cluster* algorithm, Algorithm V simply identifies contiguous groups of cells with probability greater than the user selected threshold b. However, in domains with more complex probability distributions more sophisticated clustering algorithms like the K-means algorithm or Expectation Maximization can be used to accomplish the clustering. The resulting clusters are contained in the list clusterList. Next, the elements of clusterList are prioritized. The primary criteria used for the prioritization is the amount of entropy contained in a cluster. Formally, for a cluster $C \in clusterList$ ComputeClusterPriority(C) returns $\sum_{c_i} \frac{1}{H(c_i)}$ for $c_i \in C$. Clusters exceeding the minimum priority threshold for tasking are added to $EOList$.

Algorithm 1: Algorithm to find area to image

```
AllocateSensor($P_{high}$)
(1)  $clusterList \leftarrow []$
(2)  $EOList \leftarrow []$
(3)  while true
(4)    $clusterList \leftarrow$ Cluster($P_{high}$)
(5)    foreach $cluster \in clusterList$
(6)      $cluster.priority \leftarrow$ ComputeClusterPriority($cluster$)
(7)      if $cluster.priority > MIN\_PRIORITY$
(8)        $EOList.append(cluster)$
```

Algorithm 2: Algorithm to cluster like probabilities in BBGF

```
CLUSTER($G$)
(1)  $nextID \leftarrow 0$
(2)  foreach $c \in G$
(3)    if $c.clusterId = null$
(4)      if $c.prob > b$
(5)        $c.clusterId \leftarrow nextId++$
(6)      else
(7)        $neighborList \leftarrow$ NEIGBORS($c$)
(8)        foreach $n \in neighborList$
(9)          if n.clusterId $\neq$ null
(10)           $c.clusterId \leftarrow n.clusterId$
(11)           break
```

VI. Cooperative Search

In this section, we describe the cooperative path planning for maximizing the team's expected information gain and, hence, its estimate of emitter locations.

Shortly before traversing a path, the UAV plans its next path, using an RRT planner as described below. The path is encapsulated in a token and forwarded to some of the other team members. It is not critical for the token to reach all other team members, although team performance will be better if it does. UAVs store all the paths they receive via tokens. When planning new paths a change in entropy due to other UAVs flying their planned paths is assumed by the planner. Effectively, the entropy is reduced in areas where other UAVs plan to fly, reducing the incentive for flying in those areas. If the UAV does not know the current planned path of a particular UAV, it takes the last known location of that UAV, i.e., typically the last point on the last plan from that UAV, and assumes that the UAV moved randomly from there.[c] Using this technique, the UAVs mostly search different parts of the environment, but will sometimes have overlapping paths. Importantly, the approach is computationally and communication efficient, scalable and very robust to message loss.

VI.A. Modified RRT Planner

Once the UAV has the entropy map and knowledge of the paths of other UAVs, it needs to actually plan a path that maximizes the team's information gain. We chose to apply an RRT planner[5,6] because it is fast, capable of handling large, continuous search spaces and able to handle non-trivial vehicle dynamics.

However, efficient RRT planners typically rely on using a goal destination to guide which points in the space to expand to. In this case, there is no specific goal, the UAV should just find a path that maximizes information gain. Initial tests with an RRT planner showed them to be inefficient in such cases. Moreover, the RRT planner did not handle the subtle features of the entropy map well. To make the planner more efficient for this particular problem, it was necessary to change a key step in the algorithm. Specifically, instead of picking a new point in space to expand the nearest node towards, a promising node is selected and expanded randomly outwards in a number of directions. This modified search works something like a depth first search, but with the RRT qualities of being able to quickly handle large, continuous search spaces and vehicle dynamics. Notice that this change also eliminates the most computationally expensive part of a normal RRT planner, the nearest neighbor computation, making it much faster.

Algorithm VI.A shows the modified RRT planning process. Input to the algorithm includes a cost map encoding the goals of the vehicle and another cost map with the known paths of other vehicles. Lines 1-5 initialize the algorithm, creating a priority queue (*plist*) and initial node (n). The ordering of the priority queue is very important for the functioning of the algorithm, since the highest priority node will be expanded. The function COMPUTEPRIORITY uses both the cost of the node and the number of times it has been expanded to determine a priority. Intuitively, the algorithm works best if good nodes that have not been expanded too many times previously are expanded. The main search loop is lines 6-17 and is repeated 20,000 times (about 10ms on a standard desktop.) The highest priority node is taken off the queue (then

[c]In future work, we may take into account that the other UAV will also be attempting to maximize entropy and thereby create better models of what it intends to do.

added again with new priority). This node, representing the most promising path, is expanded 10 times in the inner loop, lines 10-17. The expansion creates a new node, representing the next point on a path, extending the previous best path by a small amount. The Expand function is designed so that all new nodes lead to kinematically feasible paths. The function ComputeCost then determines the cost for the new search node, taking into account the cost of the node it succeeds and the *cost maps*. The cost map representing other paths will return positive infinity if the new node leads to a path segment that would lead to a collision. The expanded nodes are added to the priority list for possible future expansion and the process continues. Finally, the node with the lowest cost is returned. The best path is found by iterating back over the *prev* pointers from the best node.

Algorithm 3: RRT Planning Process

```
RRTPlanner($x, y, CostMaps, time, state$)
(1)  $plist \leftarrow []$
(2)  $n \leftarrow \langle x, y, t, cost = 0, prev = \emptyset, priority = 0\rangle$
(3)  $n \leftarrow$ ComputePriority($n$)
(4)  $plist.insert(n)$
(5)  $best = n$
(6)  foreach 20000
(7)    $n \leftarrow plist.removeFirst()$
(8)    $n.priority \leftarrow$ ComputePriority($n$)
(9)    $plist.insert(n)$
(10)   foreach 10
(11)     $n' \leftarrow$ Expand($n$)
(12)     $n'.prev = n$
(13)     $n'.cost =$ Cost($n, CostMaps$)
(14)     $n'.priority \leftarrow$ ComputePriority($n'$)
(15)     $plist.insert(n')$
(16)     if $n'.cost < best.cost$
(17)       $best \leftarrow n$
(18) Return $best$
```

The planning process plans several kilometres and takes less then 0.5s on a standard desktop machine, even with other proxy processes continuing in parallel.

Using the Planner If the UAV only plans a short distance ahead, it can fail to find plans that lead it to high value areas that are a long distance away. However, if the UAV plans long paths, it loses reactivity to new information (both sensor readings and plans of others). Our approach is to allow the UAV to plan long paths, but only use the first small piece of the path. In this way, the UAV will reach high value, distant areas by repeatedly creating plans to that area and executing part of the plan, but it can also react quickly to new information.

VII. Experiments

In this section, we present empirical simulation results of the approach described above. The approach is implemented with the Machinetta proxy[10] framework integrated with the Sanjaya UAV simulation environemnt. The signal model is derived from real data from an RSSI sensor flown on a real UAV. The code used is exactly the same code as being used in an ongoing flight test, with the exception of the code between the proxy and the autopilot. The simulated environment is 50km by 50km and the results below represent several hundred hours of simulated flying time, with each data point an average of ten runs. Unless otherwise stated there were four RF emitters in the environment. Each RSSI UAV had a sensor range of 2500m to 5000m. Each scenario also included an EOIR UAV which was tasked to provide video footage of the emitters. The commander interface was provided with an automatic picture order feature which tasked the UAVs once a cluster of a small enough size was produced by the RSSI UAVs. Each picture order contains the coordinates of the believed location of the emitter and these coordinates are compared against the true location. We allowed a threshold of 1000m, and picture orders outside this range of the true location were considered false positives. For picture orders inside this range, the distance from the true location as well as the average distance traveled by the RSSI UAVs up to that point were recorded. The simulator and proxies are spread out over up to 15 desktop computers and communication is via multi-cast UDP resulting in around 3% message loss. These experiments are conducted in simulation due to the practical difficulty of conducting experiments with large numbers of physical UAVs. With an industrial partner this approach was validated with four physical UAVs in a series of tests in late 2006 and early 2007.

Number of UAVs and Number of Emitters The second experiment varied both the number of emitters and number of UAVs in the environment. Figure 7 shows that more UAVs led to a faster decrease in the KL-divergence, showing that the additional UAVs were useful. Interestingly, more UAVs actually made reducing the KL-divergence faster. We hypothesize that this was because the UAVs were able to use the additional signals in the environment to quickly identify RF emitter locations.

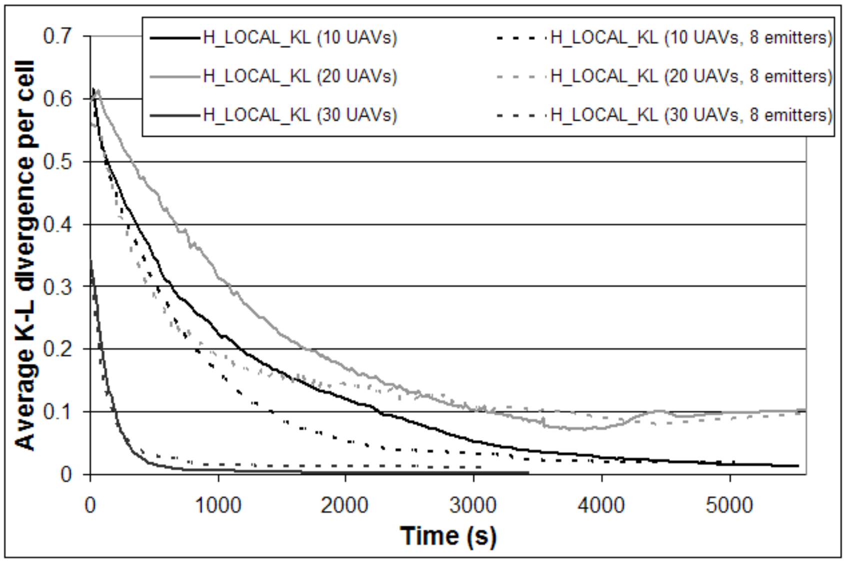

Figure 7. The impact on KL-divergence of changing the number of UAVs and the number of RF emitters.

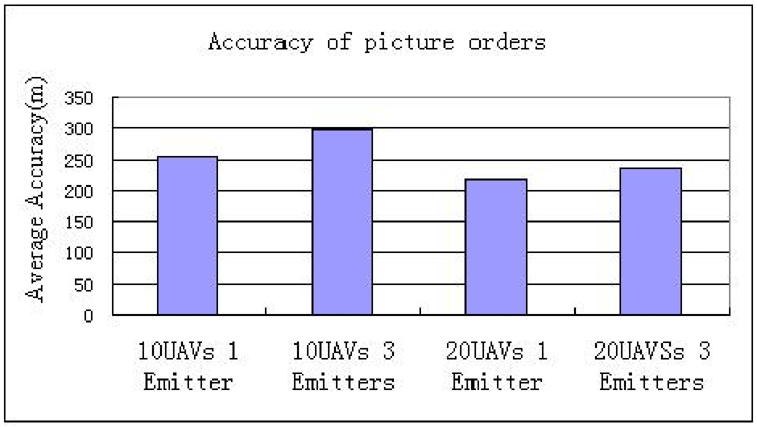

Figure 8. Average Accuracy of picture orders.

Figure 8 shows the average distance from the picture order to the true location of the target in the various configurations. It can be seen from this that increasing the number of RSSI UAVs in the scenario provides more accurate picture orders. An increase in the number of emitters caused a loss of accuracy, likely to due ambiguity in the recieved signal due to overlap.

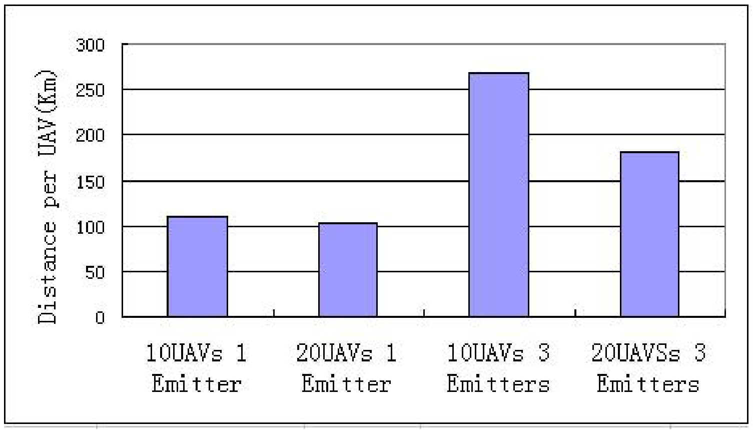

Figure 9. Distance traveled per RSSI UAV.

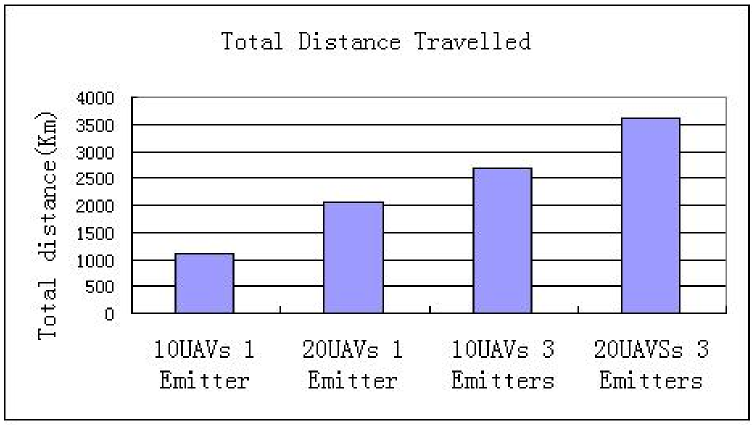

Figure 10. Total distance traveled by UAVs.

Figure 9 shows the average distance traveled per RSSI UAV in metres before all the emitters were found. Figure 10 shows the total distance traveled by all the RSSI UAVs before all emitters were found. Notice that in the 20 UAV case the UAVs traveled less distance each, but more in total than in the 10 UAV case, suggesting a speed/fuel tradeoff when determining how many UAVs to deploy.

Figure 11 shows the total number of false positives that were recorded over 10 runs for each configuration. A false positive was defined as any picture order that was not within a 1000m range of the true location. The number of false positives is very low in all cases, but higher when there are multiple emitters, pointing to the ambiguity induced by the overlapping signals.

During the initial experiments it was observed that frequently early in a run an RSSI-equipped UAV would pass near an emitter and an rough estimate of the emitter location would be obtained. Most of the entropy around the emitter would be removed leaving only a small patch of very high entropy right around the emitter. Unfortunately, on this single pass the estimate of location would not be sufficiently accurate to justify sending an EO-equipped UAV, but other areas of the environment were completely unsearched so the UAVs would fly elsewhere. Thus, the clusters would remain too large until most of the map had been visited. As an test comparison, we added a bonus reward for a UAV to visit small areas with very high entropy, expecting that this would allow these emitters to be located more quickly.

Figure 12 shows the comparison of performance with and without this extra cost map enabled on a scenario of 20 RSSI UAVs and 3 emitters. The extra cost map allows the first emitter to be discovered approximately 40% earlier as the UAVs focus on removing the entropy surrounding the target. The effect is considerably less on the time taken to find all emitters however which could be attributed to the fact that UAVs are slightly less inclined to explore, instead focusing on suspected targets. We anticipate that at even higher numbers of emitters, the impact of this additional reward may even be negative. This will be

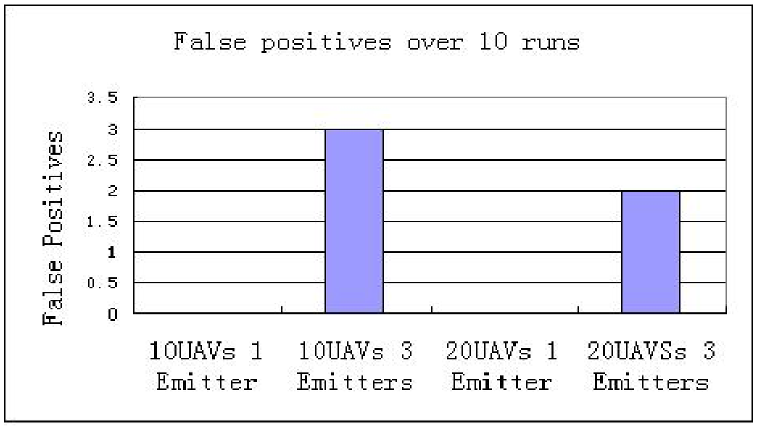

Figure 11. The number of false positives.

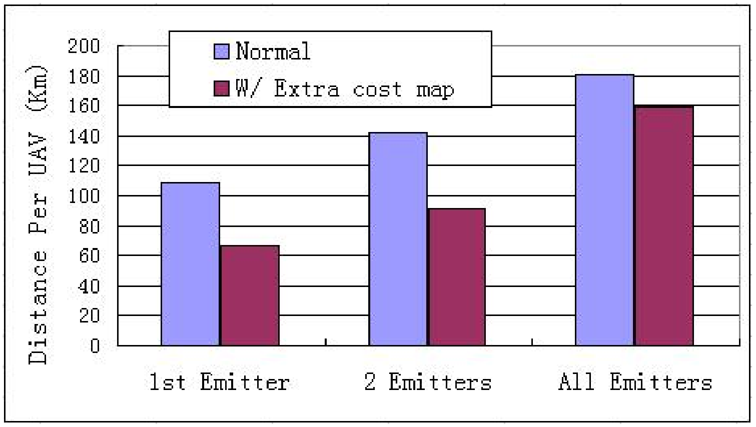

Figure 12. Comparison of performance with and without high belief cost map.

investigated in future work.

VIII. Related Work

The application presented in this chapter builds on a large body of previous work spanning a range of areas.

In[9] a method for distributed probabilistic state estimation is presented. In this work agents share local beliefs with neighbors through a query-answer protocol. There are several difficulties with this approach for our application. Firstly, for a UAV team the concept of a neighbor is problematic, since UAVs move so quickly neighbors change often and simply keeping up with who a UAVs neighbors are can be expensive. Secondly, information exchanges are strictly between pairs. This fact, in combination with the KL-divergence criteria for determining the importance of information to be shared, puts excessive responsibility on local agents to determine the importance of local information to the team. Furthermore, each shared reading is only shared with a single neighbor. This limits the potential of shared readings to contribute to the entire team and therefore limits team search optimization. In contrast, our approach enables readings with high utility to the team to propagate to the entire team. Consider for example if a UAV flies directly over an emitter, clearly such a reading should be shared with the entire team.

In[13,14] the locations of sources are detected using information theoretic techniques. This work depends on a fixed array of receivers and as such does not contend with the added complexity of incorporating moving sensors into the formulation or proactive path planning for sensors to improve source localization.

In[4] a multiple UAV team is used to localize a group of emitters. In that work, a single UAV broadcasts all sensor readings to teammates. To ameliorate the exponential cost of this sharing paradigm, UAVs form sub-teams which each maintain a separate subteam posterior. The main drawback of this approach is that it is not possible to optimize search paths over the entire team. In fact, with this approach all optimization occurs within small sub-teams.

IX. Conclusions and Future Work

This paper presented an approach to locating RF emitters with teams of UAVs. The team used RSSI sensors to roughly locate the location of the emitters then tasked EO-camera equipped UAVs to provide video streams on small areas likely to targets. While the approach was effective, we believe that it can be made more efficient with better path planning and more cooperation between UAVs.

Acknowledgements

This research has been sponsored in part by AFOSR FA9550-07-1-0039, AFOSR FA9620-01-1-0542, L3-Communications (4500257512) and NSF ITR IIS-0205526.

References

[1] R. Beard, T. Mclain, D. Nelson, and D. Kingston. Decentralized cooperative aerial surveillance using fixed-wing miniature uavs. *IEEE Proceedings: Special Issue on Multi-Robot Systems*, to appear.

[2] L. Bertuccelli and J. How. Search for dynamic targets with uncertain probability maps. In *IEEE American Control Conference*, 2006.

[3] S. Butenko, R. Murphey, and P. Pardalos, editors. *Recent Developments in Cooperative Control and Optimization.* Springer, 2003.

[4] P. DeLima, G. York, and D. Pack. Localization of ground targets using a flying sensor network. In *SUTC (1)*, pages 194–199, 2006.

[5] N. Kalra, D. Ferguson, and A. Stentz. Constrained exploration for studies in multirobot coordination. In *Proc. IEEE International Conference on Robotics and Automation*, 2006.

[6] S. LaValle and J. Kuffner. Randomized kinodynamic planning. *International Journal of Robotics Research*, 20(5):378–400, 2001.

[7] H. P. Moravec. Sensor fusion in certainty grids for mobile robots. *AI Magazine*, 9(2):61–74, 1998.

[8] D. Pack, G. York, and G. Toussaint. Localizing mobile RF targets using multiple unmanned aerial vehicles with heterogeneous sensing capabilities. In *IEEE International Conference on Networking, Sensing, and Control*, 2005.

[9] M. Rosencrantz, G. Gordon, and S. Thrun. Decentralized sensor fusion with distributed particle filters. In *Proceedings of the Conference on Uncertainty in AI (UAI)*, 2003.

[10] P. Scerri, D. V. Pynadath, L. Johnson, P. Rosenbloom, N. Schurr, M. Si, and M. Tambe. A prototype infrastructure for distributed robot-agent-person teams. In *The Second International Joint Conference on Autonomous Agents and Multiagent Systems*, 2003.

[11] T. Schouwenaars, J. P. How, and E. Feron. Multi-vehicle path planning for non-line of sight communication. In *IEEE American Control Conference*, 2006.

[12] P. Velagapudi, O. Prokopyev, K. Sycara, and P. Scerri. Maintaining shared belief in a large multiagent team. In *In Proceedings of FUSION'07*, 2007.

[13] M. Wax. *Detection and estimation of superimposed signals.* PhD thesis, Stanford Univ, 1985.

[14] M. Wax and T. Kailath. Detection of signals by information theoretic criteria. *IEEE Trans. Acoust., Speech, Signal Processing*, 33, 1985.