Locating RF Emitters with Large UAV Teams

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

April 24, 2007

Abstract

This chapter describes a principled, yet computationally efficient way for a team of UAVs with Received Signal Strength Indicator (RSSI) sensors to locate radio frequency emitting ground vehicles in a large environment. Such a capability has a range of both civilian and military applications. RSSI sensor readings are noisy and multiple emitters will cause ambiguous, overlapping signals to be received by the sensor. Generating a probability distribution over emitter locations requires integrating multiple signals from different UAVs into a Bayesian filter, hence requiring cooperation between the UAVs. To build a coherent distributed picture given communication limitations, the UAVs share only those sensor readings that induce the largest changes in their local filter. Each UAV translates its probability distribution into a map of information *entropy* and then plans a path that will maximize the reduction in entropy (or conversely provides the highest information gain.) Planned paths are shared with a subset of other UAVs to minimize overlapping search. Experiments in a medium fidelity simulation environment show the approach to be lightweight and effective. Live flight results with lightweight Class I UAVs validate our approach.

1 Introduction

The rapidly improving availability of small, unmanned aerial vehicles (UAVs) and their ever decreasing 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 meters 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 chapter, 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 UAV teams range from finding lost hikers or skiers carrying small RF beacons to military reconnaissance operations. Moreover, the core 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 technologies 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 [12] 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].

Our approach to this problem has three key elements that enable locating RF emitters with large teams of lightweight UAVs. The first key element is a distributed filter to localize RF emitters in the environment. Each UAV has a Binary Bayesian Grid Filter [7] where a value of a cell in the grid represents the probability that there is an emitter in the corresponding location on the ground. Due to limitations on available communication bandwidth, it is infeasible for UAVs to share their entire distribution, instead they share a small subset of their sensor readings with others in the team. Hence, departing from previous approaches that elicited a model of what teammates know in order to choose what to send [9], we started from the assumption that if some information leads to large local information gain, it will probably do so for much of the team. We investigated two information gain based heuristics for choosing which readings to share with teammates. The first heuristic is to send sensor readings that have the greatest impact on the UAV's local probability distribution. The second heuristic is to create a parallel probability distribution based purely on readings received from teammates and send sensor readings that have the biggest impact on that distribution. Intuitively, the first heuristic sends readings that were most important for the local UAV, while the second sends sensor readings that are most important to the team, given a local model of what the team knows. Experiments show that the first heuristic results in better team behavior than sending random messages, but the second heuristic performs worse than random.

The second element of the approach is to tightly couple estimates of the current locations of the emitters to the UAV path planning process. Specifically,

a probability distribution over emitter locations is translated into a map of the information *entropy* in the environment. UAVs plan paths through areas of maximum entropy, hence maximizing expected information gain. The UAVs plan only a relatively short distance ahead in each planning cycle. This approach allows the UAVs to be reactive to new information, which is critical when sensors are highly uncertain and the domain is dynamic. For example, if a UAV traverses an area, but the sensor readings do not provide an accurate picture of that area, the entropy will remain high and the UAV will consider re-traversing the area. Notice that the entropy map coupled with the path planner looking to maximize information gain provides an integrated way for trading off between going to the locations where there will be most information gain and locations that can be quickly reached.

The third key element of the approach is a very lightweight, computationally inexpensive method for cooperative path planning. The important application feature underlying the approach is that due to the high uncertainty and dynamicism in the environment, some overlap of paths is acceptable (or even desirable), provided that the UAVs mainly spread out and search areas of maximum entropy. Our approach is for each UAV to share its planned path with some other members of the team. When planning, each UAV estimates the change in entropy that would be induced by those paths being flown by others and plans on the resulting entropy map. If the most current path of a particular UAV is not known the most recent location is used to roughly estimate where that UAV might be searching.

2 Problem

This chapter presents a method for localizing an unknown number of RF emitters using a team of UAVs. UAVs are outfitted wth RSSI sensors which detect the power of an RF signal at a position in space. 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.[1] 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

[1] This will be relaxed in future work.

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 1 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 2 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.

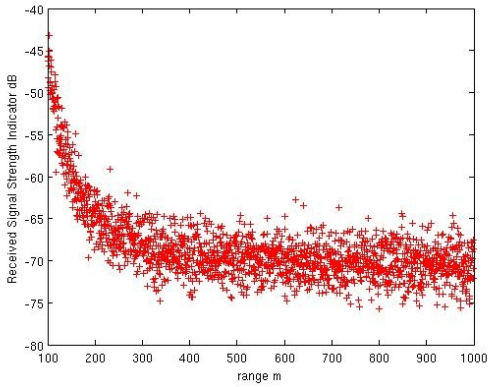

Figure 1: Sensor readings taken from different distances from an RF emitter.

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

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

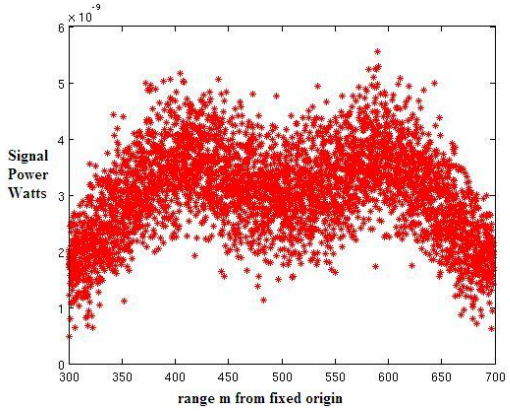

Figure 2: Sensor readings taken when flying between two emitters, first near one, then near the other.

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 Kullback Leibler divergence and $\beta_{1...3}$ are weights which control the importance of the individual factors in the optimization process.

3 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.

3.1 Implementation

The overall, integrated process aims to balance the desire to have a principled, formally grounded approach, yet be lightweight and robust enough to be practical for a team of UAVs. 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, the simulations use *exactly* the same proxy code as the live flight experiments with physical UAVs. Figure 3 shows the main components and information flows from the perspective of one UAV-proxy.

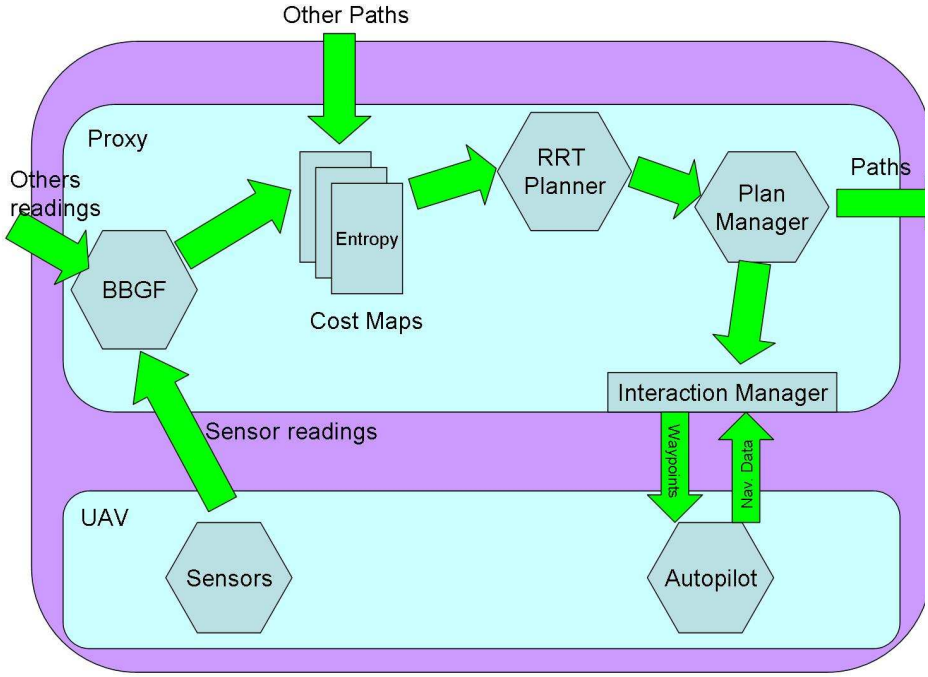

Figure 3: Block diagram of architecture.

4 Distributed State Estimation

In this section, we describe the filter used to estimate the locations of the emitters and the decisions individual UAVs make about sending information to one another.

4.1 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.[2] For a grid cell c the probability that it 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 straightforward 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 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 4 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.

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.

Entropy 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 5 shows how probability and entropy are related.

[2] 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.

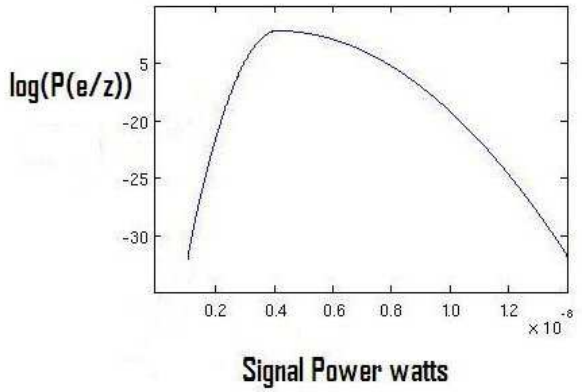

Figure 4: Mapping between probability and signal strength.

4.2 Information Sharing Approaches

For UAVs to plan the best possible paths, i.e., ones that lead to the greatest information gain for the team, it is important that each member of the team have an accurate picture of the distribution. Hence, UAVs must share local sensor readings with other members of the team. However, it is not scalable to simply send all sensor readings, nor is it likely to be particularly useful since some readings will not change the distribution very much. In this section, we describe a number of heuristics that are used to decide which sensor readings to pass around the team.

There are two reasons why we choose to share sensor readings rather than sharing probability distributions. First, for arbitrary probability distributions it is difficult to find concise representations that can be easily sent. Second, each UAV will have different confidence in different parts of its distribution and this confidence would need to be calculated and communicated with the distribution. While these problems are not insurmountable, they justify first trying the simpler approach of sending raw sensor readings.

Rosencrantz and Thrun [9] developed an approach to distributed particle filters that relied on teammates providing some information about what they know, so that the most appropriate information can be sent to each teammate. While such an approach clearly has some benefits in terms of getting the right information to the right team members, it does not scale to larger teams, from either a computational or communication perspective.

Instead, by using only local information to determine what readings to send and allowing other team members to decide whether those readings are for-

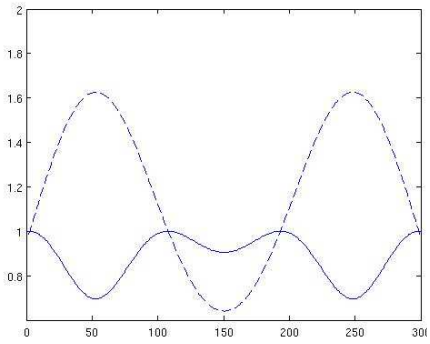

Figure 5: Mapping between probability of an emitter and entropy. The broken line shows the probability and the unbroken line the entropy.

warded, we reduce computational and communication complexity. Specifically, each agent looks at each sensor reading they get, either directly from their sensors or from other agents. If they think that this reading is sufficiently useful, they will create a *token* containing both the sensor reading and a time to live, TTL. The TTL is initially set to some small number (see 4.2.1). The token is randomly forwarded to a teammate[3]. The team member receiving the reading will integrate it into its own probability distribution. Each token has a unique identifier which is used to ensure sensor readings are only incorporated into the filter once. If the receiving agent finds the reading useful, it will increase the TTL on the token, otherwise it will decrease the TTL. While $TTL > 0$ and not all team members have been visited by the token, it will continue to be passed around the team, but as soon as $TTL = 0$ propagation stops. In this way, readings that are widely useful to the team are widely shared because many UAVs will increment the TTL, but those that are not widely useful will either be not shared at all or shared with only a small portion of the team (see 4.2.1).

The UAVs increment the TTL on tokens with sensor readings that lead to a new distribution with a KL-difference from the original distribution above some threshold α. Formally, the UAV increases the TTL on a token containing, $z_t(\vec{l})$, iff:

$$D_{\mathrm{KL}}(P_t^i(e_1 \dots e_n | z_{t_0}^i \dots z_t^i)) || P(e_1 \dots e_n | z_{t_0} \dots z_{t-1}^i)) =$$
$$\sum_i P_t^i(e_1 \dots e_n | z_{t_0}^i \dots z_t^i) \log \frac{P_t^i(e_1 \dots e_n | z_{t_0}^i \dots z_t^i)}{P_t^i(e_1 \dots e_n | z_{t_0} \dots z_{t-1}^i)} > \alpha$$

Intuitively, the UAVs are sending the most important readings from their perspective. In the results below, we refer to this heuristic as H_LOCAL_KL.

However, in some situations, information that does not seem important locally, may be important to the rest of the team. For example, a sequence of

[3]More intelligent approaches than completely random can be envisioned, but random sending minimizes computational requirements at the UAV and works effectively.

readings might slowly change the local perspective, with none of the readings having large enough KL information to send, but overall having high value. A second KL-difference heuristic utilizes a second probability distribution over emitters locations, but created *only* from sensor readings received on incoming tokens or sent on out-going tokens. Intuitively, this second distribution models the team's perspective of the BBGF. The heuristic H_TEAM_KL increases the TTL on tokens where the sensor reading leads to a KL-difference greater than the threshold on this model of the team's perspective on the BBGF. In experiments below we baseline the approach by sending random readings, denoted H_RAND.

4.2.1 Analysis

In this section we describe an analytical approach to modeling the propagation of sensor readings using the H_LOCAL_KL or H_TEAM_KL heuristics. Let p denote the probability that an agent will compute a new distribution with KL-divergence greater than the threshold α, given a new sensor reading. We assume that for a given sensor reading, p is identical for all m agents, and all agents will make a decision independently of the others. We also assume that an agent will never forward a reading to another agent that has already seen it; this can be implemented by attaching a history of recipients to the token.

Let c be the TTL increment. Because no agent ever receives the same reading twice, the total number of agents that ultimately receive a token always has the form $T = ic + 1$, where $i \in \{0, 1, 2, \dots\}$. The distribution of T for values less than m is given exactly by

$$\begin{aligned} \Pr(T = ic+1) &= p^i(1-p)^{ic-i+1} \sum_{x_1=1}^{c} \sum_{x_2=x_1+1}^{2c} \\ &\quad \cdots \sum_{x_{i-2}=x_{i-3}+1}^{(i-2)c} (i-1)c - x_{i-2} \end{aligned} \tag{1}$$

and the expected value of T can be calculated directly by

$$\begin{aligned} \langle T \rangle &= \left(\sum_{i=0}^{\lceil (m-1)/c \rceil} (ic+1) \Pr(T = ic+1) \right) \\ &\quad + m \left(1 - \sum_{i=0}^{\lceil (m-1)/c \rceil} \Pr(T = ic+1) \right) \end{aligned} \tag{2}$$

Calculating $\langle T \rangle$ from Eq. 2 can be cumbersome for large teams, but fortunately significant insight into the behavior of the system can be obtained without resorting to brute calculation. An agent receiving a sensor reading will forward it to pc other agents on average. When $pc < 1$ and $m \gg c$, $\Pr(T = m) \approx 0$ and the expected value of T can be approximated by the geometric series $\langle T \rangle \approx \sum_{j=0}^{\infty} (pc)^j = 1 + pc/(1-pc)$. When $pc > 1$, on average

each forwarding of a token will result in even more agents forwarding the sensor reading, and hence $\Pr(T = m) > 0$ even for very large m. As p increases from $1/m$ to 1, $\langle \frac{T}{m} \rangle$ increases to 1, primarily because $\Pr(T = m)$ increases toward 1. Intuitively, when $pc > 1$, if enough of the team receives a reading, it becomes very likely that eventually all of the agents will receive the reading. Mathematically this is shown by the the fact that the probability of a token stopping before reaching all of the team decreases exponentially with the accumulated TTL. The use of an initial TTL greater than 1 takes advantage of this fact and greatly increases $\langle T \rangle$ for $p > 1/c$, although it has a much lesser effect when $p < 1/c$.

The dramatic change in behavior at $p = \frac{1}{c}$ offers a promising way to choose c. Suppose that sensor readings are of two types, either useful to the team or useless to the team, and that agents correctly classify useful readings with probability p (and thus forward them) and incorrectly classify useless readings with probability $1 - q$ (and thus forward them with probability q). This causes both useful and useless readings to be forwarded through the team, and we wish to choose c such that the fraction of useful messages passed is maximized. Since the number of messages passed for a reading is equal to the number of agents that receive the reading, we wish to maximize the ratio$\langle T_p \rangle / (\langle T_p \rangle + \langle T_q \rangle)$, where T_p, T_q are the number of agents that ultimately receive useful and useless readings, respectively. As long as $p > q$, this can be accomplished by choosing c such that $q < \frac{1}{c} < p$. This is quite powerful because as long as agents are correct more often than they are wrong (a quite reasonable assumption), then $q < 1/2 < p$, and so $c = 2$ suffices to dramatically reduce the fraction of useless messages. Figure 6 shows the effect of different values of c on $\langle T_p \rangle / (\langle T_p \rangle + \langle T_q \rangle)$ for $p = 0.8$ and $q = 0.3$ and $m = 500$; for these settings the optimal choice is $c = 2$.

5 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.[4] Using this technique, the

[4] 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.

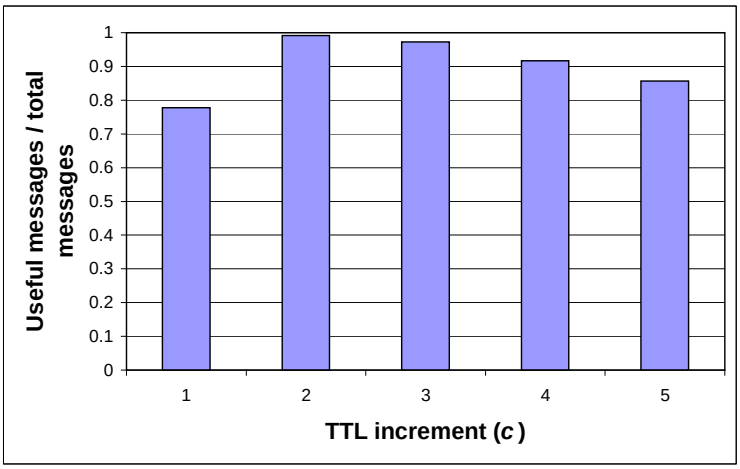

Figure 6: The ratio of useful messages to total messages for a team of 500 agents that forward useful readings with probability $p = 0.8$ and forward useless readings with probability $q = 0.3$. The fraction of useful messages is maximized at $c = 2$.

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.

5.1 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 [6, 5] 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 5.1 shows the modified RRT planning process. Input to the al-

gorithm 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 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 1: 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 kilometers 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.

6 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 either the Sanjaya UAV simulation environment or with an OPNET simulation environment. 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. Unless otherwise noted, the simulated environment is 50km by 50km and there were four RF emitters in the environment. The results below represent several hundred hours of simulated flying time, with each data point an average of five runs. 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. This approach was validated at L-3 Communications Integrated Systems in a series of live flight tests in late 2006 involving up to four Class I UAVs under autonomous control by the Machinetta proxies.

Information Sharing Experiments In the first experiment, we looked at the three different information sharing heuristics. Figure 7 shows the average KL-divergence from the ground truth over time. Ground truth is modeled as tight $\frac{1}{r^2}$ probability around the real emitter location. The figure shows that all the information sharing algorithms were effective at determining the location of the emitters, but that H_LOCAL_KL was substantially better than the other heuristics. Interestingly, sending random sensor readings, H_RAND, around the team was clearly better than H_TEAM_KL, sending readings according to a model of the team. Figure 8 gives one possible reason for this, i.e., that H_TEAM_KL sent very few readings around. H_LOCAL_KL gives a low number of messages along with its good KL-divergence, showing it to be clearly the best heuristic.

Number of UAVs and Number of Emitters The second experiment varied both the number of emitters and number of UAVs in the environment. Figure 9 shows that more UAVs led to a faster decrease in the KL-divergence, showing that the additional UAVs were useful. Interestingly, more UAVs actually

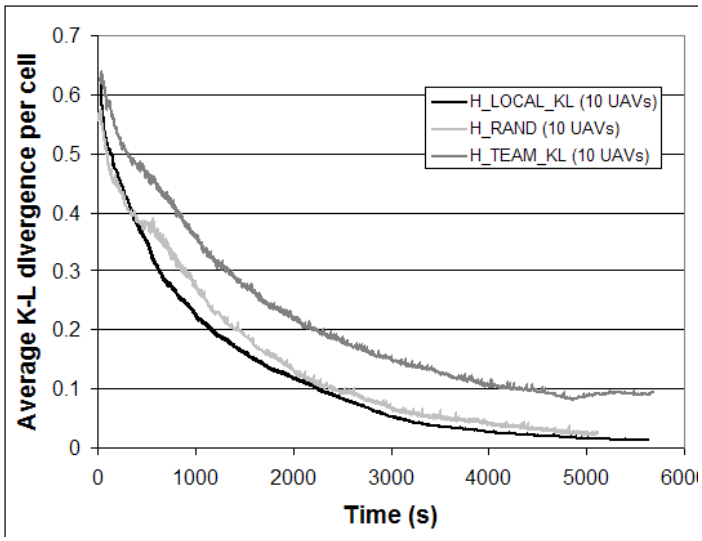

Figure 7: The KL-divergence over time for three different information sharing algorithms.

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.

Intermittent Signals The next experiment varied how often the RF emitters were giving off a signal that could be detected, see Figure 10. The four emitters had periods ranging from 5 seconds to 30 minutes, then the percentage of that period that they were on for was varied from 25% to 100%. Curiously, the KL-divergence appears better when the emitter is *off* more. However, this is only due to a quirky interaction between the KL-divergence measurement and the very noisy sensors. Specifically, the noisy sensors do not allow the UAVs to very precisely locate the emitters, so believing that they were not there at all could actually lower the KL-divergence. Figure 11 shows an oscillation in the number of messages sent between UAVs as emitters turn on and off.

Probability of Collision One of the issues that must be addressed in any practical autonomous UAV system is the possibility of collision. In this experiment, we examine how the probability of intra-system collision (collision between UAVs within the autonomous UAV system) varies with the number of UAVs. We note that our path planning approach will naturally tend to avoid collisions because of the tendency for the UAVs to spread out in order to maximize entropy gain as they coordinate their path planning. Nonetheless, because of software time delay, communications bandwidth, wind, navigation error, etc., the probability of collision is non-zero, especially if a large number of UAVs is

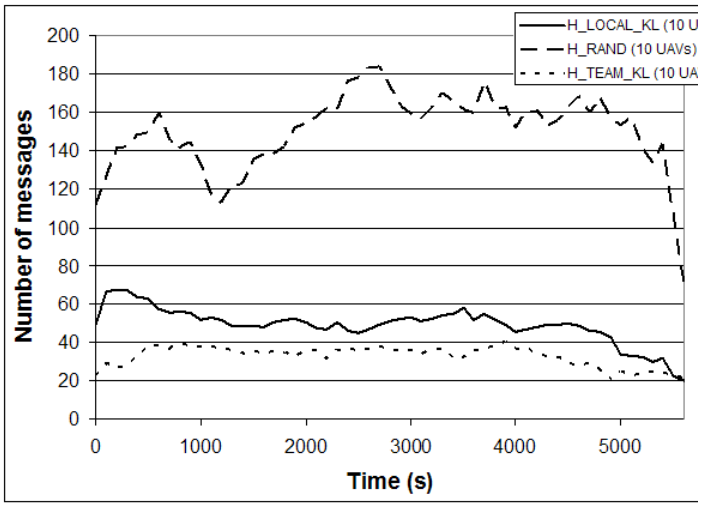

Figure 8: The number of messages sent between UAVs for three different information sharing algorithms.

covering the same region. For the simulation illustrated in Figure 12, the search area is 1 square kilometer, the UAVs are all flying at the same altitude, and simulated wind is 5 m/sec, with wind gust standard deviation of 2 m/sec. For the purposes of this simulation, a collision is defined to occur anytime the distance between a pair of UAVs is less than 5 meters. The time interval for the probability of collision metric is from the start of the mission until the entropy map is 80

Live Flight Experiments Live flight experiments with teams of up to 4 UAVs demonstrate the validity of the overall approach. The RSSI sensor, shown in Figure 13 and weighing only a few ounces, was selected to be suitable for integration on a lightweight Class I UAV, such as the Procerus UAV shown in Figure 14. In the experiments, the Procerus UAVs are hand launched, then once all are in the air, the control is handed over to the Machinetta proxies. The proxy software is hosted on a ground station rather than the on-board processor in order to minimize on-board processing requirements for the experimental system. As discussed earlier, the simulations and live flight tests use the same proxy software. This joint simulation / live flight development environment not only minimizes simulation artifacts, but also accelerates the overall development by allowing a rapid cycle of algorithm and code development, simulation, live flight tests, post-mission analysis, and back to algorithm and code development. Figure 15 illustrates some of the preliminary live flight results in a side-by-side comparison with simulation results for a 1 square kilometer region with three UAVs. The bottom half of Figure 15 shows the entropy map (left-hand side)

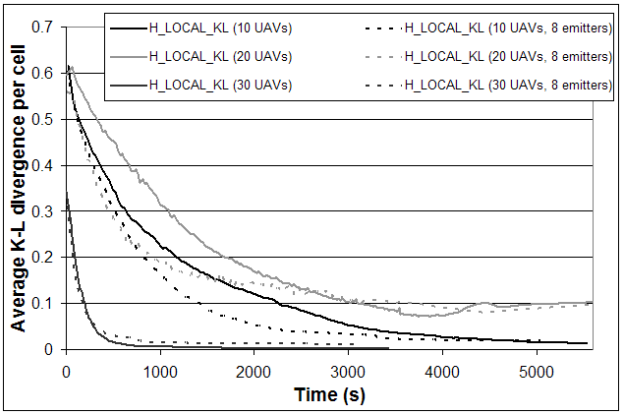

Figure 9: The impact on KL-divergence of changing the number of UAVs and the number of RF emitters.

and BBGF resulting from a live flight test at roughly the 20 minute point. The top half of Figure 15 shows the corresponding simulation results. From this, we can see that the emitter localization (shown in light green in the BBGFs on the right hand side) is, not surprisingly, somewhat better for the simulation than for the live flight test. Analysis of a series of live flight tests and simulations indicate that one significant factor is sporadic loss of communications during the live flights. The UAV autopilot is programmed to circle anytime during loss of communications in order to allow for quick visual identification of data link problems and to provide for taking manual control of the UAV if necessary. Upon restoration of communications, the autopilot automatically goes back to control of its proxy.

7 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 respon-

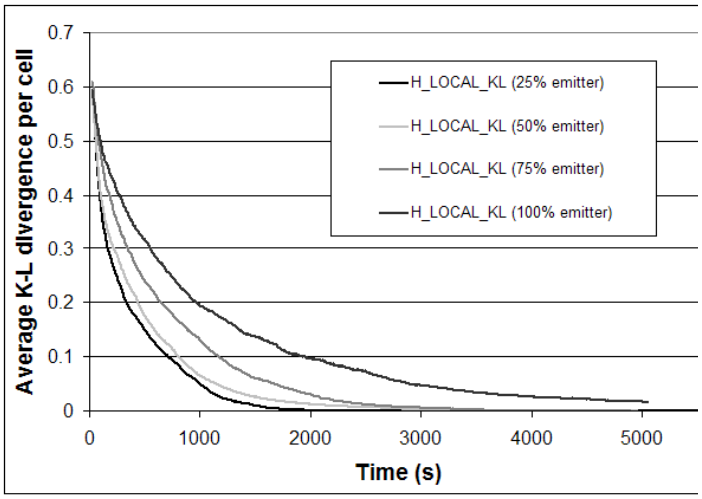

Figure 10: The KL-divergence over time for four different percentages of time the RF emitters emit.

sibility 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 [12, 13] 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.

8 Conclusions

This chapter presented an integrated approach to finding RF emitters with a large team of UAVs. Simulation and live flight experiments show the approach to be effective, light-weight and robust. The key to the scalability and robustness

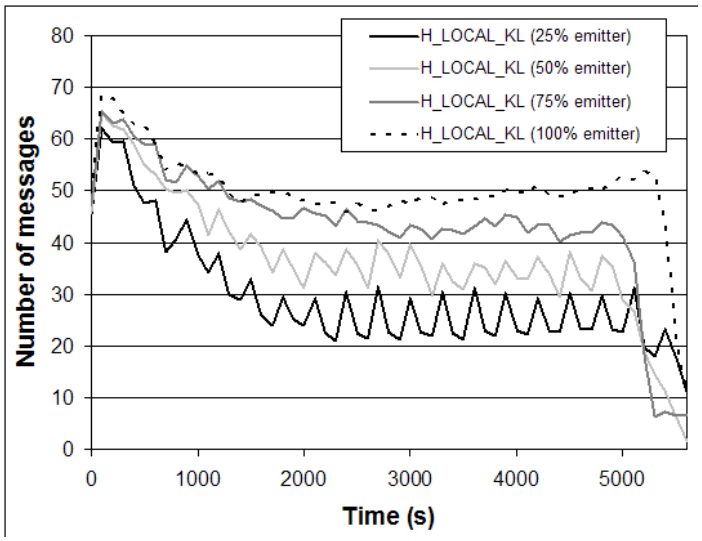

Figure 11: The number of messages over time, for different levels of intermittency.

was to find algorithms that can exploit information provided by the team but not rely on it. The most critical aspect of this was to design algorithms where local knowledge was exploited to make coordination decisions. While local knowledge was not always accurate, many local decisions make for good overall behavior because on average local decisions are good.

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.

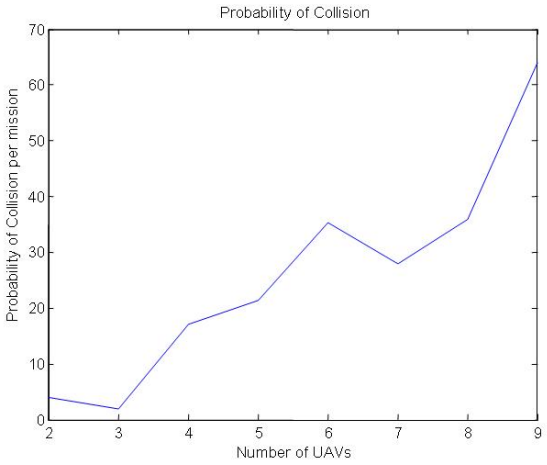

Figure 12: Probability of collision versus number of UAVs in 1 square kilometer search area.

[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.

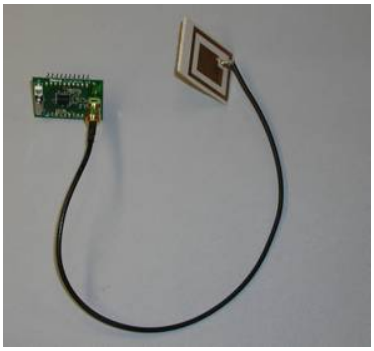

Figure 13: RSSI sensor used for live flight experiments.

Figure 14: Procerus UAV with 60" wingspan used for live flight experiments.

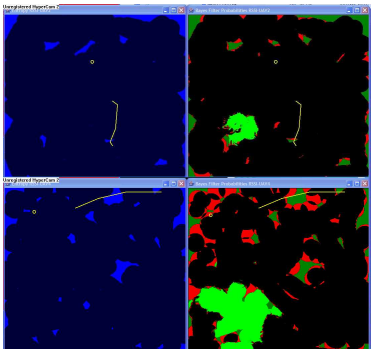

Figure 15: Example simulation and live flight comparison of entropy and BBGF maps for similar scenario.

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

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