Transitioning Multiagent Technology to UAV Applications

Paul Scerri[1], Tracy Von Gonten[2], Gerald Fudge[2], Sean Owens[1] and Katia Sycara[1]

1. Carnegie Mellon University
2. L-3 Communications / Integrated Systems

ABSTRACT

This paper describes the transition of academically developed multiagent technology for UAV coordination to an industrially developed application. The specific application is the use of lightweight UAVs with small Received Signal Strength Indicator sensors to cooperatively locate targets emitting radio frequency signals in a large area. It is shown that general techniques can be effectively transitioned, sometimes with minimal changes. However, clear differences in engineering and testing requirements of academia and commercialization require extensive effort in developing simulation and live flight testbeds. Although the technology has not yet been commercialized, initial live flight testing shows the potential of the approach.

1. 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 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 teams of very small UAVs. 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 technologies required to build a UAV team for multi-UAV applications have been developed and are reasonably mature and effective [1, 2, 6]. Moreover, intelligent agent technology appears to be a promising approach for building such applications because the abstraction is clear, extensible and powerful. However, practical use of multi-UAV teams has yet to emerge because some basic usability and engineering issues have not been adequately addressed. Key open issues include scaling the number of UAVs controllable by a single user, developing an efficient and effective transition path from research to application and development of sensors specifically appropriate for small UAVs. In this paper, we describe ongoing collaboration between academia and industrial partners aimed at transitioning state-of-the-art research into practical applications.

The academic partner in the collaboration, Carnegie Mellon University, has developed algorithms for the key UAV coordination issues involved in locating RF emitters. The approach has been presented in detail elsewhere[6] and is presented only briefly here. Binary Bayesian Grid Filters (BBGF)[5] are used 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[8]. 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. A version of a Rapidly-expanding Random Tree (RRT) planner[4] is used to determine UAV paths. Each of the influences on the planned path, i.e., entropy, the paths of others and terrain, is captured by a *cost map* which are combined and used by the RRT planner to determine the utility of many paths. Clustering algorithms on the BBGF are used 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.

The overall approach described above has been implemented within the Machinetta[7] proxy infrastructure and evaluated in increasingly high fidelity testbeds. 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. More importantly, the same code is used on low and high fidelity test beds, allowing an effective transition process. Figure 1 shows the control interface, with the map display on the left, camera view at top right (simulated, in this case) 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

Cite as: Transitioning Multiagent Technology to UAV Applications, Scerri, Von Gonten, Fudge, Owens and Sycara, *Proc. of 7th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2008)- Industry and Applications Track*, Berger, Burg, Nishiyama (eds.), May, 12-16., 2008, Estoril, Portugal, pp. XXX-XXX.

100m across.

The industrial partner in the collaboration, L-3 Communications Integrated Systems (L-3/IS), integrated the UAV system, including the Machinetta software and algorithms developed by CMU, communications interfaces, Procerus 60" UAV platforms with autopilot and camera, RSSI sensors, and ground station. To validate the behavior and robustness of the resulting system, L-3/IS developed a high fidelity simulation environment with OpNet and performed live flight experiments. The high fidelity simulation environment, besides providing for performance characterization, yielded some unexpected benefits including detecting differences in model assumptions between L-3/IS and CMU, and the ability to analyze and correct emergent behaviors (i.e., undesired system behaviors that result from complex interaction of the sub-systems and environment). With tighter FAA restrictions on live flight testing, having a high fidelity simulation environment becomes even more important.

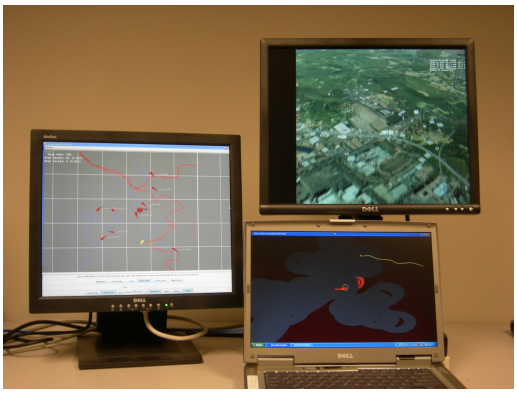

Figure 1: Operator interface station showing map view, camera view and filter output.

2. PROBLEM

Our collaborative research focused on the problem of localizing an unknown number of RF emitters using a team of UAVs. Most UAVs in this team are assumed to have RSSI sensors which measure RF signal strength at a particular frequency band. A small number of UAVs are outfitted with EO sensors capable of streaming video back to a user. We assume that the environment is too large to 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.[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 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 $P^i_t(e_1 \dots e_n | z^i_{t_0} \dots z^i_t)$.

[1]This will be relaxed in future work.

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 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:

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

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.

2.1 RSSI Sensor

RSSI measurements are available with many transceiver chips, including, for example the Chipcon CC1020 shown as an inset in Figure 2 together with the RSSI antenna and Procerus UAV that we used in our experiments. The information provided by RSSI sensors alone, however, is very low quality – although range can be approximately inferred if source strength and direction relative to the UAV antenna are known, these parameters are initally unknown and are subject to change in a dynamic environment. In addition, multipath fading and shadowing will distort the RSSI measurements – when combined with noise, the resulting RSSI measurements may have a very large variance even when range and angle are fixed. For example, Figure 3 presents RSSI measurements as a function of range taken from live flight experiments. In this case, the range is known and the UAV antenna is approximately omni-directional so that the RF emitter angle relative to the antenna response does not come into play in the RSSI measurements. This example illustrates the difficulty in relying on RSSI information from a single sensor to estimate range. Another factor that makes it difficult to interpret the information from a single RSSI sensor is the fact that there might be multiple emitters contributing to the RSSI measurements. In the context of an adaptive distributed sensing network, where spatially diverse information may be combined from a team of cooperating UAVs, RSSI sensors can be used to localize RF emitters even with all of these disadvanges.

3. ALGORITHMS

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

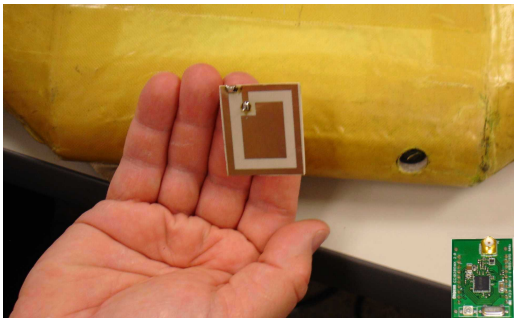

Figure 2: Procurus Close-Up with RSSI Antenna and RSSI Sensor (inset).

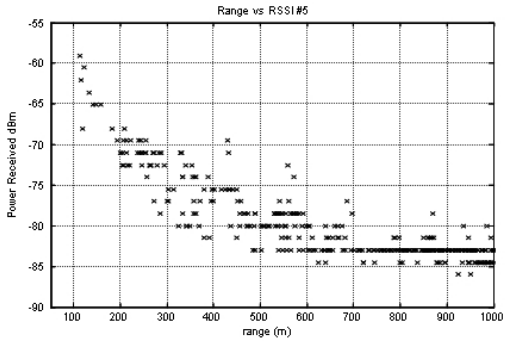

Figure 3: RSSI Measurements vs. Range (Live Flight Data).

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.

3.1 Distributed State Estimation

In this section, we describe the filter used to estimate the

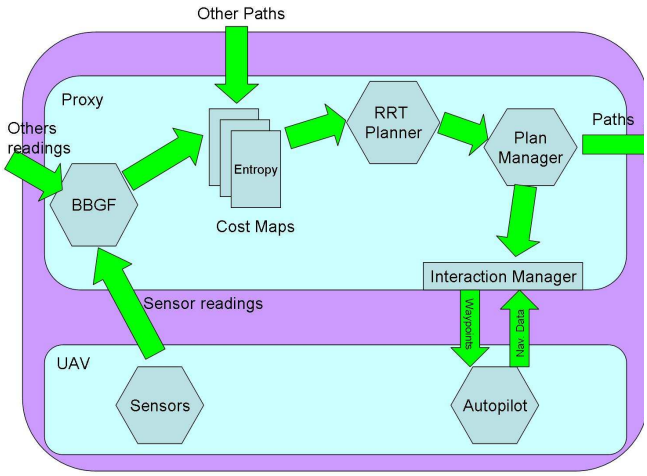

Figure 4: Block diagram of architecture.

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[8].

3.2 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 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.

Notice that there is no normalization process across the grid because the number of emitters is not known. If the

[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.

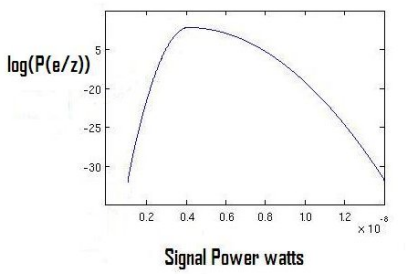

Figure 5: Mapping between probability and signal strength.

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.

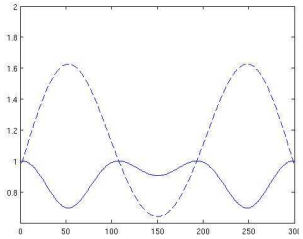

Figure 6: Mapping between probability of an emitter and entropy. The broken line shows the probability and the unbroken line the entropy.

3.3 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.[3] 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.

3.4 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 [4, 3] 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 3.4 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 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

[3]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.

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.

4. TRANSITION

The transition from pure simulation to live flight with UAVs involved a number of steps. The first step was to select a UAV platform and RSSI sensor. As discussed earlier, we integrated the Chipcon CC1020 onto Procerus UAVs with 60" wingspan. Figure 7 shows a team of four Procerus UAVs ready for flight. We selected the Procerus UAV based on: cost, built-in cameras, ease of hardware / software integration, data link which supported simultaneous multi-UAV control, and the Kestral autopilot interface. The Machinetta agents provide waypoints to the autopilot and thus do not need to have precise understanding of the UAV flight control surfaces and dynamics. This not only simplifies the autonomous control, but also allows inclusion of other platforms, including possibly large fixed wing UAVs or rotary wing UAVs.

The second step was to develop a high fidelity simulation environment that would support multiple agents running on separate processors in a distributed network similar to the UAV distributed network. The high fidelity simulation environment uses OpNet (a commercial RF / communications modeling tool) with the terrain modeling module DTED data, integrated with UAV platform dynamic models, wind models (added later based after early live flight testing), and the Terrain Integrated Rough Earth Model (TIREM) propagation delay model. TIREM predicts the RF propagation loss from 1 MHz to 40 MHz, over land and water, and includes a number of parameters in simulating RSSI measurements, including ground conductivity, humidity, and surface refractivity. We also developed interface software to link between the Machinetta agents / algorithms and the OpNet simulation. With all of this in place, we can run the Machinetta intelligent agents in a distributed network and the simulation can take into account network communications issues in addition to RF propagation, UAV dynamics, wind, terrain, and antenna response. On the Machinetta side of the simulation interface, the Machinetta software is identical to the Machinetta software used for live flight testing, thus ensuring high fidelity of the results with minimal software effort.

The third step was to validate the expected RSSI model used in the simulations. After integration of the RSSI sensors onto the UAVs and basic check-out, we flew three UAVs in close formation with a narrow-band signal on the ground and compared the RSSI measurements – this provided an estimate of the variance of the individual RSSI sensor in a live flight environment and this also provided a validation that the RSSI sensor readings between different UAVs were consistent.

The fourth step was to develop additional layers of software to ensure positive control of the UAVs in case of undesired emergent behaviors. This extra software checked the waypoints generated by the Machinetta agents to ensure the waypoints were within the operational area; in addition, this software allowed for full override of the Machinetta agents at any time. In the case of communications loss, the Kestral autopilot would force the UAV to circle the last known waypoint.

Some additional items that were addressed during integration and transition to live flight included: Re-host of software from Linux OS to Windows OS to support our available distributed network compute platforms; translation of coordinate systems between the autopilot local East, North, Up system and the Machinetta Euclidean coordinate system; signal level units conversions to dBm with approximate gain of antenna / receiver; and modification of network communications from the UDP Machinetta default to TCP/IP when interfacing outside of Machinetta.

One of the advantages of Machinetta is that it is a very general framework and is thus quite flexible. Thus, the overall adaptation from prior multi-agent applications to the RF emitter localization UAV team domain did not require any significant changes to the core Machinetta software. In addition, adding the cooperative team behavior to perform video search of the expected emitter locations was also performed with minimal or no impact to the core Machinetta software. On the other hand, this level of abstraction can be a potential difficulty as seen by our team in tracking down some differences in the signal model assumptions – Machinetta is written in fairly abstract objected-oriented code and tracking down variables can be difficult if one is not familiar with the code.

5. COMPARING RESULTS

RSSI Localization Live Flights and Emergent Behavior.

Figure 7: Procerus UAV Team (on Rack Before Flight Test).

Our initial live flights tested the RSSI localization adaptive team distributed sensing behavior. Prior to live flights we ran Monte Carlo simulations to characterize the performance, due to time and expense of flights. Live flights are used to validate the Monte Carlo results, including any specific unusual behaviors observed in simulations. Our RSSI localization live flight tests qualitatively agreed with our Monte Carlo simulations performed using the high fidelity OpNet simulation environment, but both cases (live flight and OpNet simulation) exhibited some behavior not observed in the CMU simulation environment. One behavior in particular illustrates the potential of undesired emergent behaviors– the occasional circling of a UAV about a single waypoint, as can be seen by the yellow UAV path in Figure 8. This particular phenomena was caused by a combination of factors related to interaction of different system components, including the Machinetta update rate and the interface software between Machinetta and the UAVs. Machinetta was generating updates at a very high rate and sometimes waypoint commands for different UAVs would collide because of the finite transmission time to the UAV. When these messages collided, some of the waypoints would be dropped and once the UAV reached its waypoint, it would circle about that waypoint while waiting for an update. This behavior was also observed in the high fidelity OpNet simulation environment for two reasons: the same interface software was used and the agents were in a physically distributed network with the possibility of message collisions. Later flight tests are planned to test additional behaviors such as the video search team behavior after RSSI localization.

Although correcting the circling around a waypoint problem was fairly straightforward, this case does illustrate the potential of undesired emergent behaviors and how important it is to incorporate constraints on these behaviors. In this case, one can imagine alternative designs that would have resulted in a much worse emergent behavior. One option would be for a UAV to default to maintain its course and speed when no new messages were received. In a very early simulation, a different emergent behavior resulted in the UAVs leaving the area completely, so our research team was deliberate in preventing any emergent behavior that would lead to the UAVs leaving the area. Another alternative would be if Machinetta generated low-level auto-pilot control commands – dropped messages under this scenario could easily result in the UAV crashing. While these are rather obvious poor design choices to avoid, this example shows how an emergent behavior with potentially disastrous consequences can develop in a complex system and how emergent behaviors can be constrained through design choices.

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 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.

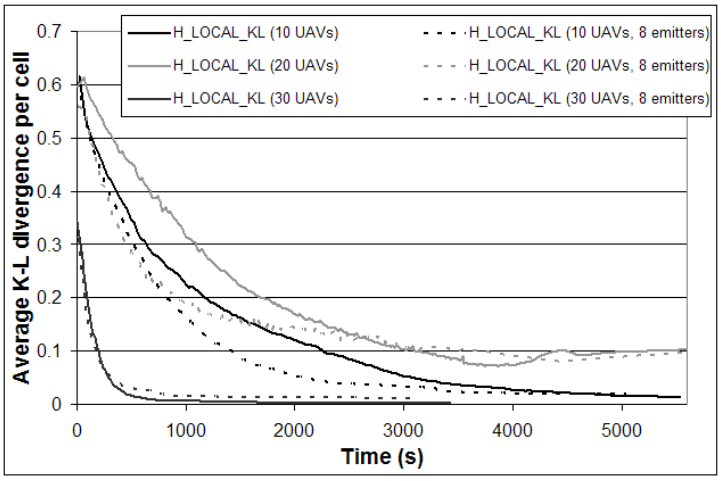

Figure 9: The impact on KL-divergence of changing the number of UAVs and the number of RF emitters.

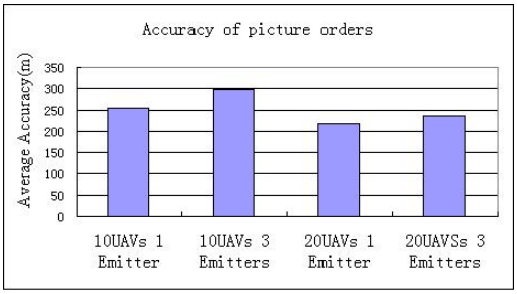

Figure 10: Average accuracy of picture orders.

Figure 10 shows the average distance from the location where a UAV was ordered to take a picture to the true location of the target in the various configurations. Figure 10 shows 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 due to ambiguity in the received signal due to overlap.

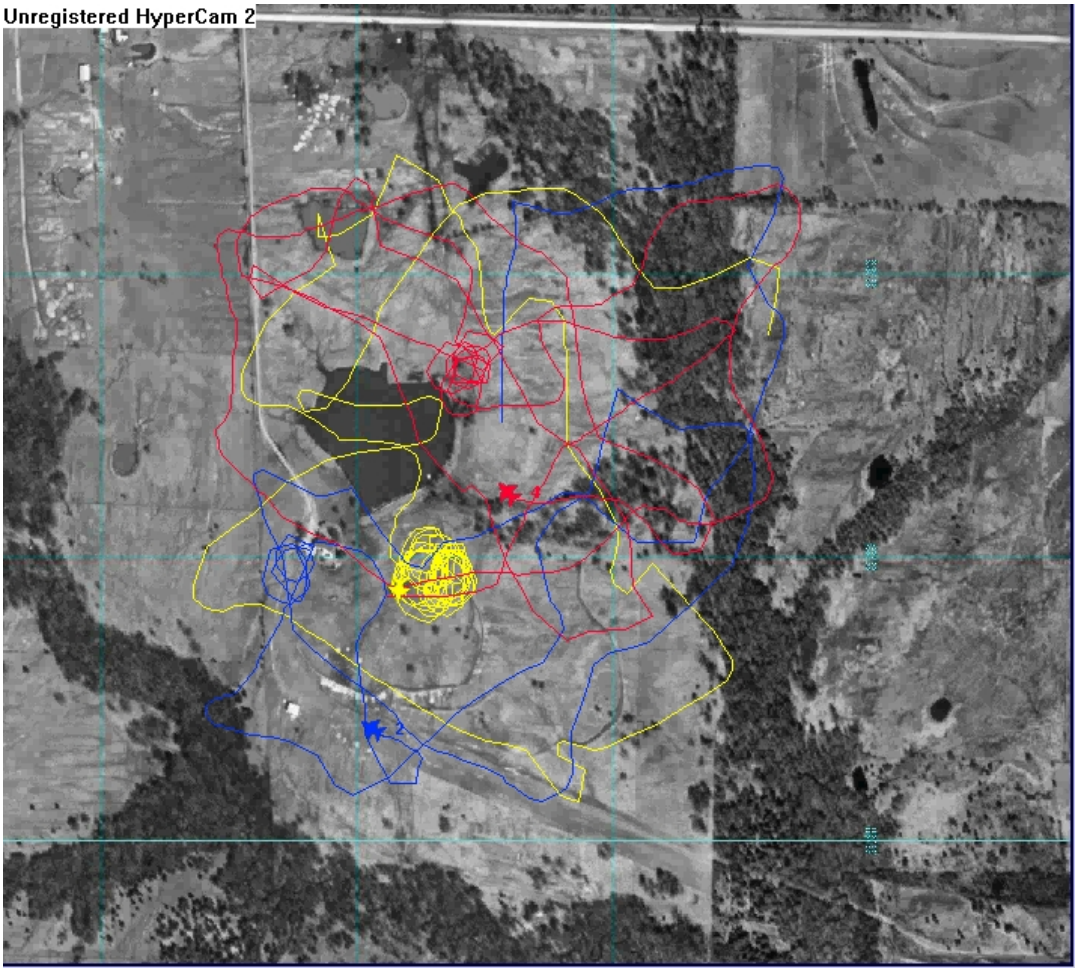

Figure 8: Example Live Flight Constrained Emergent Behavior.

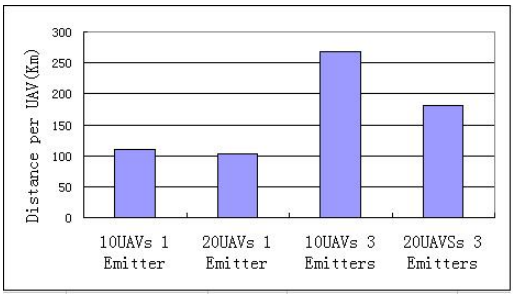

Figure 11: Distance traveled per RSSI UAV.

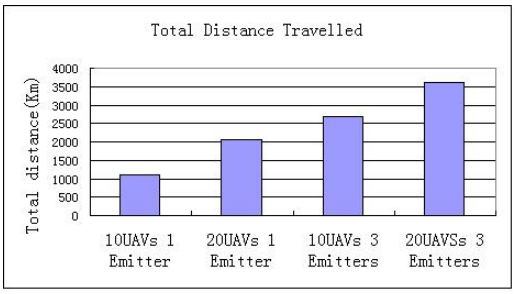

Figure 12: Total distance traveled by UAVs.

Figure 11 shows the average distance traveled per RSSI UAV in kilometers before all the emitters were found. Figure 12 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 13 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 zero when there is only one emitter and in all cases, but significant when there are multiple emitters, pointing to the ambiguity induced by the overlapping signals.

During the initial experiments it was observed frequently that early in a run a RSSI-equipped UAV would pass near an emitter and a 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

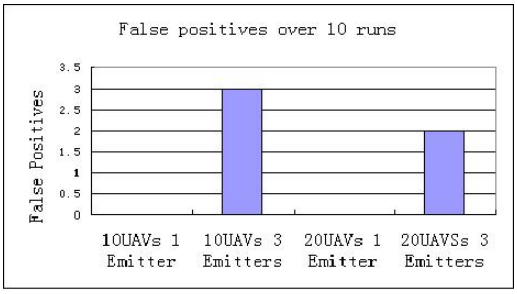

Figure 13: The number of false positives.

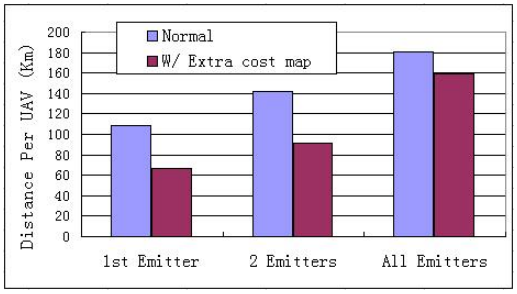

Figure 14: Comparison of performance with and without high belief cost map.

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 a 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 14 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 investigated in future work.

6. LESSONS LEARNED

The collaborative partnership between CMU and L-3/IS has proven fruitful – the combination of theory and systems integration experience allowed us to rapidly develop an autonomous UAV system far more quickly than either could have done without collaboration. In working with academia, L-3/IS has observed that university collaborators are far more willing to tackle high risk projects than typical sub-contractors; we have also found that having high level discussion of project goals instead of a detailed requirements specifications on research projects is beneficial – this avoids accidentally constraining the solution space and provides the university the flexibility to work out of the box.

7. CONCLUSIONS

This paper presented a successful transition of maturing agent technology to an industrial application. The key lesson from this collaboration, for future acedemia-industry collaboration is that the collaboration works best when tasks are divided according to the strengths of the partners, but close cooperation ensures each partner is cognizant of the other partners constraints. CMU delivered software developed in simulation and based on solid theory but only effective for the limited enviroment conditions modeled in a lightweight simulator. L-3/IS brought engineering expertise to bear to ground models, integrate with hardware and adapt high fidelity simulations. It is hoped that this approach will lead to exciting commercial applications in an effective way.

8. ACKNOWLEDGEMENTS

The authors would like to thank the following research collaborators on the project who supported system integration and flight test: Josh Anderson, Vaughn Bowman, Jon Brown, Michael Custer, Dan Rutherford, and Ken Stroud.

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

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

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

[6] P. Scerri, R. Glinton, S. Owens, D. Scerri, and K. Sycara. Geolocation of rf emitters by many uavs. In *AIAA Infotech@Aerospace 2007 Conference and Exhibit*, 2007.

[7] 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.

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