Multiple Event Localization in a Sparse Acoustic Sensor Network Using UAVs as Data Mules

Jason T. Isaacs*, Sriram Venkateswaran*, Joao Hespanha*, Upamanyu Madhow*,
Jerry Burman†, and Tien Pham‡
*Department of Electrical and Computer Engineering
University of California, Santa Barbara, CA 93106-9560
Email: {jtisaacs, sriram, hespanha, madhow}@ece.ucsb.edu
†Teledyne Scientific Co.
1049 Camino Dos Rios, Thousand Oaks, CA 91360
Email: jburman@teledyne-si.com
‡U.S. Army Research Lab.
2800 Powder Mill Road, Adelphi, MD 20783-1193
Email: tien.pham1.civ@mail.mil

***Abstract*—We report on a field demonstration of autonomous detection, localization, and verification of multiple acoustic events using sparsely deployed unattended ground sensors, unmanned aerial vehicles (UAV) as data mules, and a ground control interface. A novel algorithm is demonstrated to address the problem of multiple event acoustic source localization in the presence of false and missed detections. We also demonstrate an algorithm to route a UAV equipped with a radio to collect data from sparsely deployed ground sensors that takes advantage of the communication range of the aircraft while adhering to kinematic constraints of the UAV. A second UAV was utilized to provide video verification of localized events to a human operator at a ground control station.**

I. Introduction

Monitoring large areas and localizing events of interest is a fundamental problem with a variety of applications such as homeland security and environmental studies. Typically, localizing an event requires information from only a few sensors: for example, we can localize a source if we have access to the time of arrival (ToA) information from three sensors. However, gathering information from spatially dispersed sensors poses a significant challenge. When the sensors are placed on the ground and do not have line-of-sight links to one another, their communication range is drastically reduced, leading to disconnected networks with sparse sensor deployments. One approach to this problem is to overdeploy sensors by increasing the sensor density until the network becomes connected and the sensors can exchange information via multihop networking. However, this approach does not scale well. When monitoring large regions, it would lead to an extraordinarily large number of sensors, which is expensive and not fundamentally necessary for the purpose of event localization. We explore an alternative approach to alleviate this issue by considering a deployment with "just enough" sensors to localize the sources of interest, leading to a disconnected network. We then use an Unmanned Aerial Vehicle (UAV) as a data mule to gather data from the different sensors and make inferences regarding the event locations on the fly. In this paper, we validate this architecture by presenting results from a field demonstration where we localize multiple acoustic sources using ToA sensors and a mule-UAV whose route is optimized to quickly gather data.

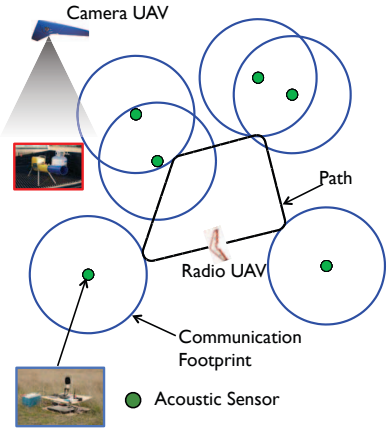


Fig. 1. Schematic of the system used in the demonstration.

This work was supported by the Institute for Collaborative Biotechnologies through grant W911NF-09-0001 from the U.S. Army Research Office. The content of the information does not necessarily reflect the position or the policy of the Government, and no official endorsement should be inferred.

978-1-4673-4941-3/12/$31.00 ©2012 IEEE

A schematic of the system used in the deployment is shown in Figure 1. We deploy six ToA sensors over a region that is roughly 1.3 km $\times$ 0.5 km in size. We use GPS receivers at each sensor to estimate their locations and synchronize them in time. Two propane cannons that have acoustic characteristics similar to artillery are fired randomly and potentially close to one another in time. A UAV flies a traveling salesman tour over the sensors, gathering ToAs and inferring possible event locations. When the inference algorithm has sufficient confidence in a candidate event, it dispatches a second UAV, fitted with a gimbaled camera, to fly over the estimated location and image the source. The data gathering and event imaging is done continuously, with the events being imaged on a first come first served basis.

There are two key technical challenges that we had to solve in the process of demonstrating the proposed system; (a) The propagation delays from different events to a sensor are different. Therefore, when multiple events happen close to one another in time, the varying propagation delays can cause the events to arrive in different orders at the sensors. Consequently, simple rules to group the ToAs from an event at different sensors, such as sorting the ToAs in ascending order and picking the ith ToA from each sensor to localize the ith event, fail. Trying all possible ToA groupings might work but is too complex: with E events and N sensors, we need to try E^N combinations. (b) The problem of planning the shortest path for a UAV to visit a set of ground sensors is combinatorial in nature. Most existing solutions make simplifying assumptions about UAV kinematics and thus often times result in planned paths that are difficult if not impossible for the UAV to follow. Additionally, most path planning solutions treat sensors as points in space and fail to take advantage of regions where ground sensor to UAV communication is possible.

We solve the problem of localizing multiple closely spaced(in time) events by parallelizing the evidence: we hypothesize discrete times at which events occur, allowing us to generate a small set of event candidates. We then choose a subset of the candidates that best explains the observations at the different sensors in an efficient manner by solving a matching problem on a graph. We use a sampling based method to solve the mule-UAV path planning problem. Sample UAV poses are used to generate a Generalized Traveling Salesman Problem (GTPS) on a directed graph, and a series of graph transformations are used to convert the GTSP to an Asymmetric TSP. During the demonstration, we found that events were quickly localized with little spatial error (within 15m and within 3 min after their occurrence), thereby illustrating the efficiency, robustness, and low-complexity of the algorithms.

Related Work: There is a rich body of work on source localization which is surveyed in [1]. Most of this research is restricted to the case when there is a single event and therefore, cannot be used in our deployment with multiple cannons. However, there is one exception: [2] develops an algorithm for localizing multiple events from their ToAs at different sensors. But this algorithm does not account for all the constraints of the localization problem and can, therefore, return more events than the number that actually occurred.

Acoustic sensor networks have been used in previous localization applications [2], [3]. In [2] a network of ToA sensors detect and localize a sniper based on the muzzle blast and shock wave. However, the nodes in [2] are deployed densely on a smaller spatial scale, so that they can form a connected network to exchange data thus avoiding the need for a data mule. A data mule architecture to transport data in a disconnected sensor network was proposed, and scaling laws for this setting were investigated in [4] and in several other papers on delay-tolerant networking, including [5]–[7]. The DTSPN for the data mule UAV seeks to combine the Dubins Traveling Salesman Problem [8] with the Traveling Salesman with Neighborhoods Problem [9]. The path planning algorithm used to address the DTSPN for the UAV data mule is most similar to the sampling based method from [10].

We considered similar sparse acoustic sensor network deployments in [3], [11]. The focus there was on UAV routing algorithms that balance the information gathered and the time taken along any route. However, this work focused on the problem of localizing events sufficiently separated in time, and the routing algorithm did not account for the kinematic constraints associated with UAVs. The algorithms that we explain briefly in this paper are explained in much more detail in [12]–[14], but the results there are restricted to simulations and do not include any details of the experiments that are the focus of this paper.

Hardware description: We used two Zon Mark IV propane cannons to create acoustic events that were similar to live artillery. We used an omnidirectional Samson C03 microphone to record the cannon shots and a Dell Latitude E5500 laptop to process the recordings. The processing involved matched filtering the data against a pre-recorded template of a cannon shot to estimate ToAs at the sensors. The laptop was interfaced to a Garmin 18-LVC GPS to provide us with the sensor's location and ensure time-synchronization across the network. Finally, each laptop was connected to a Microhard radio to forward the ToAs to the mule-UAV. Two Procerus Unicorn UAVs were used with different payloads. The imaging-UAV was equipped with a gimbaled camera. The mule-UAV forwarded the ToAs to the base-station which ran the multiple event localization algorithm and instructed the imaging-UAV to fly to estimated event positions for visual verification. We describe the hardware components in more detail in [3].

Organization: The remainder of the paper is organized as follows. We describe the system model and describe

an algorithm to address multiple event localization in Section II. In Section III, the Dubins Traveling Salesman Problem with Neighborhoods is described and a path planning algorithm is described that is particularly useful when the regions overlap. We present results from a field test in Section IV and conclude in Section V.

II. Multiple Event Localization Algorithm

We begin by providing an overview of the algorithm described in [12], [13] and then explain the modifications needed for the demonstration. Suppose that E events occur over a time window $[0, T]$ and produce ToAs at N sensors. The ToA produced by the e^{th} event, parametrized by its time of occurrence t_e and spatial location $\boldsymbol{\varphi}_e$, at sensor s is given by

$$\tau_{es} = t_e + \|\boldsymbol{\varphi}_e - \boldsymbol{\gamma}_s\| + \tilde{n} \tag{1}$$

where $\boldsymbol{\gamma}_s$ is the location of sensor s and $\tilde{n}$ is the measurement noise, distributed as $N(0, \tilde{\sigma}^2)$. We have normalized units of distance and time so that the speed of sound is 1. For each event, a sensor makes a measurement of this form with probability $1 - p_{miss}$ and completely misses it with probability p_{miss}. Additionally, each sensor also makes outlier ToA measurements, produced by "small-scale" events (say, a slamming car door). Such small-scale events are heard only at one sensor and therefore, cannot be localized. However, we must take care to discard outlier measurements while estimating event locations.

There are two key ideas behind the algorithm: first, we quickly narrow down the set of candidate events by using some, but not all, constraints of the problem. We then exploit the reduction in the number of candidates to use the remaining constraints and pick out the true events from the candidate list in a principled fashion. We explain the algorithm briefly.

Stage 1: If an event that occurred at time u produces a ToA τ at sensor s, the event location $\boldsymbol{\varphi}$ is constrained to a circle (neglecting measurement noise):

$$\|\boldsymbol{\varphi} - \boldsymbol{\gamma}_s\| = \tau - u.$$

Therefore, if we hypothesize that the ith ToA at sensor s and the jth ToA at sensor s' are produced by the same event that occurred at time u, the event location $\boldsymbol{\varphi}$ must lie at the intersection of the circles

$$\|\boldsymbol{\varphi} - \boldsymbol{\gamma}_s\| = \tau_s(i) - u \qquad \text{and} \qquad \|\boldsymbol{\varphi} - \boldsymbol{\gamma}_{s'}\| = \tau_{s'}(j) - u.$$

The points of intersection of two circles are extremely easy to compute and can, in fact, be specified in closed-form. Thus, by hypothesizing discrete event times $(\ldots, -2\epsilon, -\epsilon, 0, \epsilon, 2\epsilon, \ldots)$ and considering different pairs of ToAs, we can quickly generate candidate events.

The geometry of this processing for a hypothesized event time u is shown in Figure 2. We generate candidates close to actual event locations by intersecting circles corresponding to ToAs produced by the same event (say, red event). In this example, one such candidate is denoted by $\hat{\mathcal{E}}$. However, we also generate a number of "phantom" candidates (no event occurred there) by intersecting circles corresponding to different events (p_2, p_3, p_6 and p_7) or by intersecting circles belonging to the same event, but with a wrong hypothesized time (for example, if the green event does not occur at u, then p_4 and p_5 are phantoms). The goal of the rest of the algorithm is to discard the phantoms using measurements at all the sensors and only retain the true events.

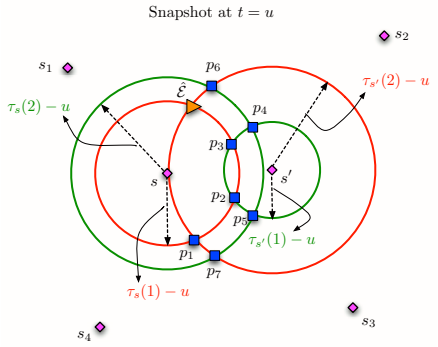


Fig. 2. Two events "red" and "green" that produce ToAs at sensors $s, s', s_1, s_2, s_3, s_4$. The red event produces ToAs $\{\tau_s(1), \tau_{s'}(2)\}$ at sensors s and s' and the green event produces ToAs $\{\tau_s(2), \tau_{s'}(1)\}$. We hypothesize an event time u and draw circles corresponding to the different ToAs. The time at which the red event occurred is close to u and hence we obtain an estimate of the red event location $\hat{\mathcal{E}}$. However, we also generate phantom estimates $p_1 - p_7$

We do this in multiple steps. First, we discard "obvious" phantoms by developing a statistical goodness metric for each event and discarding events whose goodness falls below a threshold. We also merge "duplicate" candidates to further prune the overcomplete set. We omit a description of these stages here and refer to [12], [13] for more details. At the end of this process, we have event estimates and non-obvious phantoms (candidate events which did not occur but can be explained by some observations) in the list of candidates.

Stage 2: The candidates in Stage 1 are obtained by intersecting circles drawn at a pair of sensors and do not use the information available at the other $N-2$ sensors. In Stage 2, we refine each estimate, in an iterative fashion, by using the ToAs from all the sensors. We refer to [12], [13] for more details. At the end of this stage, the estimates of the events are near-perfect, but the list of candidates still includes non-obvious phantoms.

Stage 3: We prune the remaining phantoms using a variant of the matching problem on a bipartite graph, where we pair a subset of the candidate events with the observations at the different sensors. We need to make two sets of decisions: (a) for each event in the overcomplete set, we need to decide whether it truly happened or if it is a phantom. An example of such decisions is shown in Figure 3, where the events that we declared to have occurred are shown in green and the ones we declare

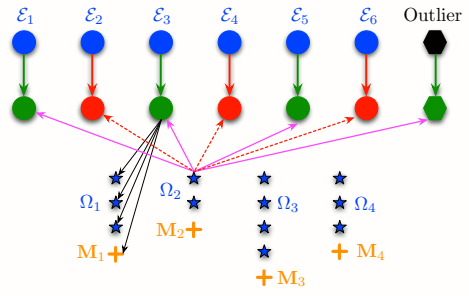


Fig. 3. Events at the output of stage 2 are shown as blue circles in the first row. The observations are denoted by blue stars, with the observations at each sensor arranged in a column. Green circles represent events that we declared to have occurred and red circles denote phantom events. We need to draw edges between the picked events and the observations, subject to constraints, so as to minimize the costs of the edges.

to be phantoms are shown in red. (b) We then establish a correspondence between the chosen events and the observations (which event produced which ToA?) subject to two types of constraints: *Event node constraints:* At each sensor, an event that occurred must produce an observation or must be missed. Thus, exactly one of the black edges that connect $\mathcal{E}_3$ to the observations at sensor 1 or the miss node M_1 must be "active". *Observation node constraints:* Each observation must be produced by a *picked* event or must be an outlier. Thus, exactly one of the pink edges that connect the observation at sensor 2 to events/outliers must be active.

Activating an edge comes with a cost: for example, the cost of declaring an event to be missed at any sensor is $\log p_{miss}$. The costs of other edges can be specified in a similar fashion, and we refer to [12], [13] for more details. All the decisions made in this process are binary valued. Thus, the overall problem can be specified as a binary integer program (maximize a linear objective function where the decision variables are all either 0 or 1). Since solving a binary integer program has prohibitive complexity, we relax the problem and solve it as linear program (variables can take any value between 0 and 1). Typically, we find that, even with the relaxation, the decision variables only take the values 0 or 1 with a large enough sensor deployment.

Modifications for the demonstration: The algorithm presented above makes two simplifying assumptions that need to be removed before we can use it in the proposed system. First, it assumes that we have access to *all* the the measurements from *all* the sensors before we start attempting localization. This does not hold when the UAV picks up ToAs sequentially: at any given time, we only have access to the ToAs produced by an event from those sensors that the UAV has visited *after* the event occurred (+ propagation delay). Second, it assumes that the ToAs produced by events from the window $[0, T]$ have been separated from the rest of the ToAs. In practice, events occur continuously, and it is unclear how to split the ToAs in this fashion (was a ToA $T + h, h > 0$ produced by an event in $[0, T]$ with a "large" propagation delay or by one in $[T, 2T]$ with a smaller delay?).

We solve both of these problems by keeping track of the last time at which the UAV visited each of the sensors. When the current time is t, let $T_i(t)$ denote the last time when the UAV was within the radio range of sensor i ($T_i(t) \leq t$). To avoid penalizing events, some of whose ToAs have not been collected, we add an "Early" node at each sensor to the graph in Figure 3 (not shown). Let E_i denote the early node at sensor i. For a candidate event (φ, u), denote its expected ToA at sensor i as $\tau_i = u + \|\varphi - \gamma_i\|$. If $T_i(t) < \tau_i$, we assign a very small cost (close to zero) to the edge between the event and the early node E_i. Thus, the event can be explained by activating the edge to E_i without paying a penalty. However, if $T_i(t) > \tau_i$, the UAV has picked up all the available information about the event, and the edge connecting the event to E_i should not be necessary. In this case, we assign a large cost to the edge to ensure that it will not be used.

For the algorithm to operate in a continuous fashion, we retain a ToA and consider it for processing until we are certain that either (a) we have gathered all the information about an event that produced it, or (b) it is an outlier. Suppose that the LP activates the edge between a candidate event (φ, u) and the ToA τ at sensor i. We check if the UAV has visited *every* sensor after the predicted ToA due to (φ, u) at each of these sensors. If so, we do not consider the ToA τ for further processing. If not, there is more information to be gathered, and we retain τ for use in further processing. If τ is an outlier ToA, the LP will not associate any candidate event with it. We purge it as follows: if the UAV has visited all of the sensors after $\tau + D$, where D is the diameter of the deployment region, and the LP still does not associate the ToA τ to any candidate, it must be an outlier and can be removed from further consideration.

III. UAV Path Planning

The acoustic unattended ground sensors are capable of detecting events at much greater distances than they can communicate with low power radios. A UAV is used to collect measurements from a sparse deployment of acoustic sensors. The algorithm used for UAV path planning is largely based on the one described in [14]. We first describe the assumptions and ideas behind the algorithm in [14] and then explain the modifications required for the real-time demonstration.

The kinematics of the UAV can be approximated by the Dubins vehicle in the plane. The pose of the Dubins vehicle $\mathbf{X}$ can be represented by the triplet $(x, y, \theta) \in SE(2)$, where $(x, y) \in \mathbb{R}^2$ define the position of the vehicle in the plane and $\theta \in \mathbb{S}^1$ defines the heading of the vehicle. The vehicle kinematics are then written as,

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \nu \cos(\theta) \\ \nu \sin(\theta) \\ \frac{\nu}{\rho} \bar{u} \end{bmatrix}, \tag{2}$$

where ν is the forward speed of the vehicle, ρ is the minimum turning radius, and $\bar{u} \in [-1, 1]$ is the bounded control input. Let $\mathcal{C}_\rho : SE(2) \times SE(2) \to \mathbb{R}_+$ associate the length $\mathcal{C}_\rho(\mathbf{X}_1, \mathbf{X}_2)$ of the minimum length path from an initial pose $\mathbf{X}_1$ of the Dubins vehicle to a final pose $\mathbf{X}_2$, subject to the kinematic constraints in (2). This length, which we will refer to as the Dubins distance from $\mathbf{X}_1$ to $\mathbf{X}_2$, can be computed in constant time [15].

Let $\mathcal{R} = \{\mathcal{R}_1, \mathcal{R}_2, ..., \mathcal{R}_n\}$ be a set of n compact regions in a compact region $\mathcal{Q} \subset \mathbb{R}^2$, and let $\mathbf{\Sigma} = (\sigma_1, \sigma_2, \ldots, \sigma_n)$ be an ordered permutation of $\{1, \ldots, n\}$. Define a projection from $SE(2)$ to $\mathbb{R}^2$ as $\mathcal{P} : SE(2) \to \mathbb{R}^2$, i.e. $\mathcal{P}(\mathbf{X}) = \begin{bmatrix} x & y \end{bmatrix}^T$, and let $\mathbf{P}_i$ be a point in $SE(2)$ whose projection lies in $\mathcal{R}_i$. We denote the vector created by stacking all n configurations $\mathbf{P}_i$ as $\mathbf{P} \in SE(2)^n$.

The DTSPN involves finding the minimum length tour in which the Dubins vehicle visits each region in $\mathcal{R}$ while obeying the kinematic constraints of (2). This is an optimization over all possible permutations $\mathbf{\Sigma}$ and configurations $\mathbf{P}$. Stated more formally:

Problem 3.1 (DTSPN):

$$\begin{aligned} \underset{\mathbf{\Sigma}, \mathbf{P}}{\text{minimize}} \quad & \mathcal{C}_\rho(\mathbf{P}_{\sigma_n}, \mathbf{P}_{\sigma_1}) + \sum_{i=1}^{n-1} \mathcal{C}_\rho(\mathbf{P}_{\sigma_i}, \mathbf{P}_{\sigma_{i+1}}) \\ \text{subject to} \quad & \mathcal{P}(\mathbf{P}_i) \in \mathcal{R}_i,\ i = 1, \ldots, n. \end{aligned}$$

We present an algorithm to address this problem which involves generating a set of $m \geq n$ sample configurations $\mathbf{S}_i \in SE(2)$, $\mathcal{S} := \{\mathbf{S}_1, \ldots, \mathbf{S}_m\}$ such that

$$\mathcal{P}(\mathbf{S}_k) \in \bigcup_{i=1}^{n} \mathcal{R}_i,\ k = 1, \ldots, m, \tag{3}$$

and $\forall i\ \exists k$ s.t. $\mathcal{P}(\mathbf{S}_k) \in \mathcal{R}_i$. The algorithm approximates Problem 3.1 by finding the best sample configurations $\mathbf{P} \subseteq \mathcal{S}$ and the order $\mathbf{\Sigma}$ in which to visit them.

Problem 3.2 (Sampled DTSPN):

$$\begin{aligned} \underset{\mathbf{\Sigma}, \mathbf{P}}{\text{minimize}} \quad & \mathcal{C}_\rho(\mathbf{P}_{\sigma_n}, \mathbf{P}_{\sigma_1}) + \sum_{i=1}^{n-1} \mathcal{C}_\rho(\mathbf{P}_{\sigma_i}, \mathbf{P}_{\sigma_{i+1}}) \\ \text{subject to} \quad & \mathbf{P}_i \in \mathcal{S} \\ & \mathcal{P}(\mathbf{P}_i) \in \mathcal{R}_i,\ i = 1, \ldots, n. \end{aligned}$$

Problem 3.2 can now be converted to a Generalized Traveling Salesman Problem (GTSP) with overlapping nodesets by sampling regions $\mathcal{R}$ with a finite set of m Dubins vehicle configurations $\mathcal{S}$. The GTSP is then transformed into a standard TSP through the Noon and Bean transformation [16]. To solve for the tours we used the symmetric TSP solver linkern available at [17], which uses the Chained Lin-Kernighan Heuristic from [18].

The GTSP can be described with a directed graph with nodes $\mathcal{N}$ and arcs $\mathcal{A}$ where the nodes are members of predefined nodesets $\mathcal{V}$. Here each node represents a sample pose from $\mathcal{S}$, and the arc connecting node $\mathbf{S}_i$ to node $\mathbf{S}_j$ represents the length of the minimum length path for a Dubins vehicle $c_{i,j} = \mathcal{C}_\rho(\mathbf{S}_i, \mathbf{S}_j)$ from pose $\mathbf{S}_i$ to pose $\mathbf{S}_j$. The nodeset $\mathcal{V}_k$ corresponding to region $\mathcal{R}_k$ contains all samples whose projections lie in $\mathcal{R}_k$, $\mathcal{V}_k := \{\mathbf{S}_i \in \mathcal{S} \mid \mathcal{P}(\mathbf{S}_i) \in \mathcal{R}_k\}$ for $i \in \{1, 2, \ldots, m\}$. The objective of the GTSP is to find a minimum cost cycle passing through each nodeset exactly one time.

Noon and Bean Transformation: What follows is a brief summary of the Noon-Bean transformation from [16] as it is used in this work. The transformation is best described in three stages.

The first stage converts the GTSP to a GTSP with mutually exclusive nodesets. This is done by first eliminating any arcs from $\mathcal{A}$ that do not enter at least one new nodeset. Next, a finite cost $\alpha \geq \sum_{(i,j) \in \mathcal{A}} c_{i,j}$ is added to each arc cost for each new nodeset the arc enters. Next, any nodes that belong to more than one nodeset are duplicated and placed in different nodesets so as to allow each node to have membership in only one nodeset. Any arcs to and from the original nodes are duplicated as well. In addition, zero cost arcs are added between all the spawned nodes of each multiple membership node. The large cost α added to all the other arcs ensures that all spawned nodes will be visited consecutively, if at all.

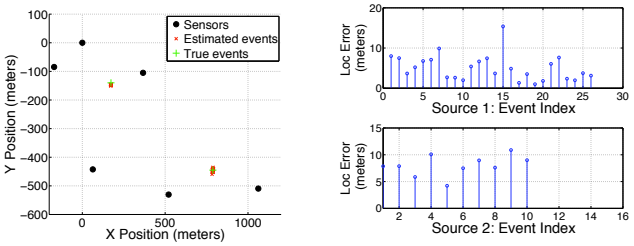
The second stage takes the GTSP with mutually exclusive nodesets and eliminates any intraset arcs, leaving a GTSP in "canonical form." The third stage of the transformation converts the canonical GTSP to a "clustered" TSP as follows. The nodes in each nodeset are first enumerated. Then, a zero cost cycle is created for each nodeset by adding zero cost edges between consecutive nodes in each nodeset and connecting the first node to the last. The *interset* edges are then shifted so they emanate from the previous node in its cycle. Finally, the clustered TSP is converted to an ATSP by adding a finite cost $\beta \geq \sum_{(i,j) \in \mathcal{A}} c_{i,j}$ to each intercluster arc cost.

Modifications for the demonstration: The DTSPN algorithm from above was modified slightly to allow for waypoint control of the UAV as well as to be more robust to disturbances such as wind. The first modification of the routing algorithm reduced the size of the communication regions in the optimization to ensure that the resulting path would penetrate the original communication region. The second modification involved sampling the desired path to obtain a sequence of waypoints to send to the UAV autopilot.

IV. Results

We deployed six ToA sensors over a 1.3 km $\times$ 0.45 km region shown in Figure 4(a). We created acoustic events using two propane cannons. The first cannon, located at $(788.3, -445)$m in Figure 4(a), was fired 26 times, and the second cannon, placed at $(173, -140)$m, was fired 10 times. The event estimates are plotted in red in Figure 4(a), and the localization errors are shown

in Figure 4(b). The localization errors are less than 16 meters on all the occasions (average errors are 5.05m and 7.98m respectively), demonstrating the efficacy of the localization scheme. Additionally, there were numerous outlier ToAs that the algorithm rejected efficiently.



(a) Event locations and estimates (b) Localization errors

Fig. 4. Results of the multiple event localization algorithm. The true and estimated event locations are shown on the left and the localization errors are shown on the right.

The route flown by the mule-UAV and the communication regions used in the DTSPN path planning algorithm are shown in Figure 5. It took on average two minutes and fifty seconds for the mule-UAV to complete the circuit and collect measurements from all ground sensors. This time is conservative due to the modifications to the algorithm that ensure that the UAV enters into each communication region (radius $= 200$m).

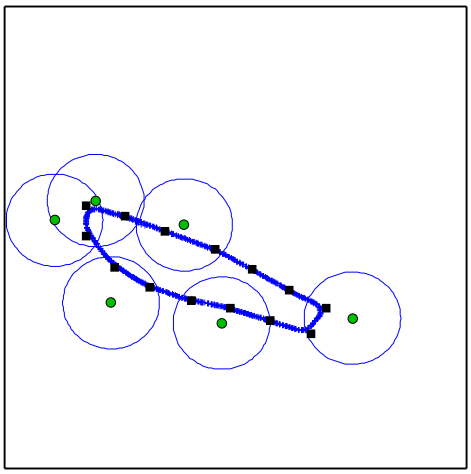


Fig. 5. Path taken by mule-UAV during tests. The desired path was sent to autopilot via square waypoints. The sensors and communication regions are represented by green and blue circles respectively.

V. Conclusions

We have presented the results of a field demonstration that utilized UAV data mules in conjunction with sparsely deployed ground sensors to detect, localize, and verify multiple acoustic events. Algorithms for multiple event localization and UAV path planning were combined, and their effectiveness was validated as events were localized quickly and accurately. A potential next step is to consider scaling to larger coverage areas where it would be beneficial to coordinate multiple mule-UAVs. Another direction of research interest considers fusing measurements from heterogeneous ground sensors for multiple source localization.

References

[1] G. Mao, B. Fidan, and B. D. O. Anderson, "Wireless sensor network localization techniques," *Computer Networks*, vol. 51, no. 10, pp. 2529–2553, July 2007.

[2] G. Simon, M. Maróti, A. Lédeczi, G. Balogh, B. Kusy, A. Nádas, G. Pap, J. Sallai, and K. Frampton, "Sensor network-based countersniper system," in *ACM Conference on Embedded Networked Sensor Systems*, Baltimore, MD, November 2004, pp. 1–12.

[3] D. J. Klein, S. Venkateswaran, J. T. Isaacs, T. Pham, J. Burman, J. Hespanha, and U. Madhow, "Source localization in a sparse acoustic sensor network using UAVs as information seeking data mules," *ACM Transactions on Sensor Networks*, vol. 9, no. 4, November 2013, to appear.

[4] R. Shah, S. Roy, S. Jain, and W. Brunette, "Data MULEs: Modeling a three-tier architecture for sparse sensor networks," in *IEEE International Workshop on Sensor Network Protocols and Applications*, Anchorage, AK, May 2003, pp. 30–41.

[5] D. Henkel and T. X. Brown, "On controlled node mobility in delay-tolerant networks of unmanned aerial vehicles," in *International Symposium on Advance Radio Technolgoies*, Boulder, CO, March 2005, pp. 7–9.

[6] D. Henkel, C. Dixon, J. Elston, and T. X. Brown, "A reliable sensor data collection network using unmanned aircraft," in *International Workshop on Multi-hop Ad Hoc Networks: From Theory to Reality*, Florence, Italy, May 2006, pp. 125–127.

[7] D. Bhadauria, O. Tekdas, and V. Isler, "Robotic data mules for collecting data over sparse sensor fields," *Journal of Field Robotics*, vol. 28, no. 3, pp. 388–404, May/June 2011.

[8] K. Savla, E. Frazzoli, and F. Bullo, "Traveling salesperson problems for the Dubins vehicle," *IEEE Transactions on Automatic Control*, vol. 53, no. 6, pp. 1378–1391, July 2008.

[9] A. Dumitrescu and J. S. B. Mitchell, "Approximation algorithms for TSP with neighborhoods in the plane," *Journal of Algorithms*, vol. 48, no. 1, pp. 135 – 159, August 2003.

[10] K. J. Obermeyer, P. Oberlin, and S. Darbha, "Sampling-based roadmap methods for a visual reconnaissance UAV," in *AIAA Conference on Guidance, Navigation, and Control*, Toronto, ON, Canada, August 2010.

[11] D. J. Klein, J. Schweikl, J. T. Isaacs, and J. P. Hespanha, "On UAV routing protocols for sparse sensor data exfiltration," in *American Control Conference*, Baltimore, MD, June 2010, pp. 6494–6500.

[12] S. Venkateswaran and U. Madhow, "Space-time localization using times of arrival," in *Allerton Conference on Communication, Control, and Computing*, Monticello, IL, September 2011, pp. 1544–1551.

[13] ——, "Localizing multiple events using times of arrival: a parallelized, hierarchical approach to the association problem," in *IEEE Transactions on Signal Processing*, 2012, to appear.

[14] J. T. Isaacs, D. J. Klein, and J. P. Hespanha, "Algorithms for the traveling salesman problem with neighborhoods involving a Dubins vehicle," in *American Control Conference*, San Francisco, CA, June 2011, pp. 1704–1709.

[15] A. M. Shkel and V. Lumelsky, "Classification of the Dubins set," *Robotics and Autonomous Systems*, vol. 34, no. 4, pp. 179–202, March 2001.

[16] C. E. Noon and J. C. Bean, "An efficient transformation of the generalized traveling salesman problem," Department of Industrial and Operations Engineering, University of Michigan, Ann Arbor, Tech. Rep. 89-36, 1989.

[17] D. Applegate, R. Bixby, V. Chvátal, and W. Cook, "Concorde TSP solver," Website: http//www.tsp.gatech.edu/concorde.

[18] D. Applegate, W. Cook, and A. Rohe, "Chained Lin-Kernighan for large traveling salesman problems," *INFORMS Journal on Computing*, vol. 15, no. 1, pp. 82–92, November 2003.