3-D Exploration with an Air-Ground Robotic System

Jonathan Butzke†, Andrew Dornbush†, Maxim Likhachev†

***Abstract*— Exploration of unknown environments is an important aspect to fielding teams of robots. Without the ability to determine on their own where to go in the environment, the full potential of robotic teams is limited to the abilities of human operators to deploy them for search and rescue, mapping, or other tasks that are predicated on gaining knowledge from the environment. This is of particular importance in real-world 3-Dimensional (3-D) environments where simple planar assumptions can lead to incomplete exploration, for example, real-world environments have areas underneath overhangs or inside caves. As an additional challenge, when the teams of robots have vastly different capabilities, the planning system must take those into account to efficiently utilize the available assets. In this paper, we present a combined air-ground system for conducting 3-D exploration in cluttered environments. We first describe the hardware and software components of the system. We then present our algorithm for planning 3-D goal locations for a heterogeneous team of robots to efficiently explore a previously unknown environment and demonstrate its applicability in real-world experiments.**

I. Introduction

Exploration of unknown environments is a cornerstone of robotic systems wishing to operate in the real-world without having constant human operator input. The ability to seek out locations to gain information about the environment is a foundation for the coverage problem and plays a significant role in tasks such as search and rescue, infrastructure inspections, and other tasks requiring a sensor to be repeatedly positioned and re-positioned so as to evaluate every possible location in the environment.

In many real-world environments a ground robot is incapable of reaching and observing all desired points. For example, small wheeled robots have difficulty seeing the items on top of a table. In these cases, a flying robot can get a camera into a better position than the ground robot can. However, since aerial robots have to support all of their weight through the expenditure of energy, they are typically limited to small payloads and short mission durations. Ground robots typically have greater power reserves and are frequently capable of long duration missions while carrying significant amounts of payload. For the environments where an aerial vehicle is required to reach certain sections, a good compromise is to search the accessible areas with a ground vehicle and reserve the aerial vehicle for just those areas that require the higher vantage point or are otherwise unviewable by the ground robot.

Thank you to Brian MacAllister for his work preparing, configuring, and operating the robots as well as for all his support during our demonstrations. This research was sponsored by USMC, contract #FA8750-13-C-0156.

† Search-based Planning Lab, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, USA {jbutzke, andrewpd}@andrew.cmu.edu, maxim@cs.cmu.edu

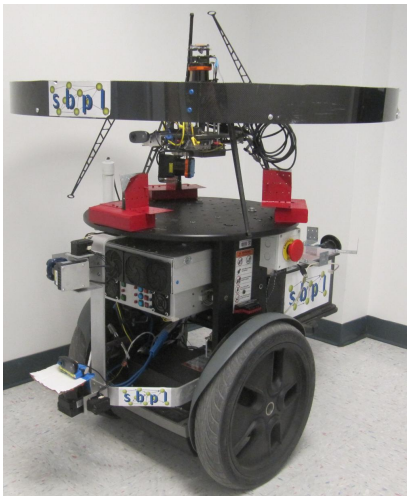


Fig. 1: UAV mounted on UGV.

Motivated by this, we present a heterogeneous system of air and ground robots (Fig. 1) that can fully explore an environment even in the case where either vehicle alone would not succeed. We also present a 3-D exploration planning algorithm that is capable of accounting for the differences in sensing and movement between the robots on the team during the goal selection process, enabling them to find a desired object within an initially unknown 3-D environment.

The paper is organized as follows: we present the state-of-the-art in exploration in Section II, our system in Section III, and our 3-D exploration planning algorithm in Section IV. Finally, we cover experimental results from our air-ground robotic system operating in an indoor environment when tasked with finding a particular object in Section V.

II. Related Work

Frontier-based exploration uses the concept of a "frontier" between the known and unknown portions of an environment and directs robots to this frontier to discover unknown areas [1]. This approach has been expanded to cover multi-robot teams [2], [3], [4], [5], [6] with great success and this work is an extension of our earlier work in this category [7], [8]. However, most of these approaches, including our previous work, only deal with 2-dimensional (planar) exploration. Even the approaches that use 3-dimensional motions tend to treat the environment more as 2.5-D (elevation- or height-map) rather than full 3-D [9], [10], [11]. In particular, few of these approaches consider the search target to be on the underside of obstacles or require movement under obstacles in order to get into a position to see the target.

One impressive approach that does consider flying under or through obstacles is the potential field/harmonic function approach used in [12]. Like our approach they use an octree to represent the environment, and a camera to explore, but where they differ is that all of their simulated UAV's were identical. It is unclear how easy it would be to incorporate a heterogeneous team of robots (including ground robots) to their scenario. Additionally, they only consider exploration of the top surface of obstacles and have no method for finding a search target located on the underside of an obstacle.

There have also been numerous works relating to cooperation between aerial and ground vehicles [13], [14], [15]. Our work is based on the general principles found in these - minimize the amount of information that has to be shared between platforms, maximize the amount of computing that can be performed in a distributed manner, and reduce the load on the operator. Our system is not fully decentralized, however; it does require a globally knowledgeable planner. While many of these only look at general collaboration between air and ground robots, our approach looks specifically at collaborative exploration of cluttered 3-D spaces.

III. Air-Ground Robotic System for 3-D Exploration

We concentrate on the problem of autonomous exploration for the purpose of finding an object of interest (OOI). The robotic team will have minimal human input: the human operator will initiate the exploration, concur with launching the UAV, and concur with any OOI detections, but will not provide any other direction or guidance to the robots[1]. Therefore, all other navigation, sensing, and decision making must be done on-board. To achieve this, we developed several modules to guide the robots through the environment as shown in Fig. 2.

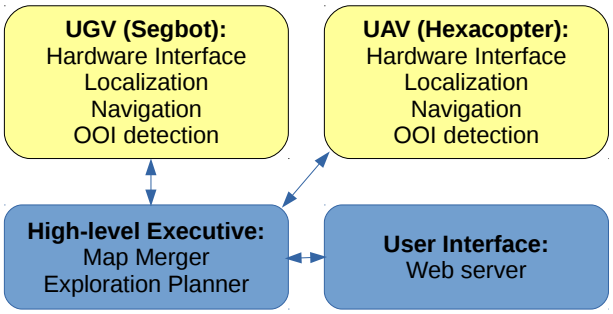


Fig. 2: Overview of system. The Robots both have sufficient computing capability and sensors to move through the environment based on higher-level goals without further guidance.

Our system uses two primary components: a team of robots executing their own software for localization, navigation, object detection, and other local processes; and a set of high-level software modules that combine the individual robot maps, select goal locations for each robot, and interface with the user.

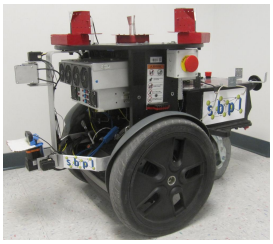
The first part is our two robots: an Unmanned Aerial Vehicle (UAV) and an Unmanned Ground Vehicle (UGV). The UGV has a large battery capacity (sufficient for 3-4 hours of operation), substantial on-board computing power, and is very stable. On the other hand, the UAV has a limited flight time (10 minutes maximum) and computing power, but can traverse terrain that the UGV cannot. In addition, it can move vertically allowing it to get a better vantage point of the environment and "see" areas the UGV cannot. The robots are detailed in Section III-A and their on-board software in Section III-B.

The second part of the system is the high-level modules which are responsible for coordinating activities between the two robots and can be executed from any available computing platform. To perform their coordination function, the high-level executive has access to a map merge capability and the exploration planner. The map merger receives map updates from the two robots and forms a global map. Using this map, the exploration planner determines appropriate goals for each robot and provides them to the executive module for transmission to the robots. In addition, the high-level software incorporates a mechanism for providing feedback to, and input from, the human operator. The high-level software modules are discussed in Section III-C.

A. Robot Platforms

1) Melvin the Segbot: The ground vehicle component of our system is Melvin the Segbot. Melvin is a custom designed robot built on a Segway RMP 200 base. Attached to this base are two computers, two 30m Hokuyo scanning laser range finders, and a Logitech webcam (see Fig. 3a).

The computing power is split between two distinct hardware components. The first is the controller computer. This machine is responsible for the direct planning and control of Melvin and is a 3.0GHz i5 with 8GB RAM. This machine handles all hardware interfaces including the motion interface to the base. The second computer is a dual quad-core Xeon server with 16GB of RAM that executes all of the high level planning including the exploration planner and map merger nodes.



(a) UGV

(b) UAV

Fig. 3: Robots used for experiments. a) UGV - Large box on right is main battery pack. Upper gray box is server, lower gray box is controller computer. Hokuyos and camera are mounted on structure on left. b) UAV - closeup showing bottom panning vertical Lidar, upper fixed horizontal Lidar, and forward facing camera.

[1]As part of a separate line of research regarding the User Interface, we imposed the constraint that the human would have to confirm OOI detections. Because of this, we included a decoy object in the environment during testing.

2) Hexacopter: The Hexacopter serves as the aerial component of the exploration team. Like Melvin, it is equipped with two 30m Hokuyo scanning laser range finders, a Logitech webcam, and its own computer. The computer is an i7-2660 with 16GB of RAM that runs all of the automation onboard. The body of the Hexacopter is a modified Mikrokopter Hexa XL frame with custom sensor and computing mounts, power distribution electronics, and blade guards (see Fig. 3b).

B. Local Software

1) General Software & Communication: All of the computers run Kubuntu 12.04 with ROS Groovy. Each robot has its own instantiation of a roscore with an additional roscore for the exploration planner and map merger, and one for the user interface. Both of these additional roscores are physically executed on the UGV server computing system. For the few messages that needed to be passed between systems, we used the ROCON software package to transfer standard ROS messages to multiple roscores. This setup was made to allow for movement of the high-level modules to any available computing system. By having its own roscore, all that was necessary was to update the ROCON links if we ran it on a different physical machine.

2) Localization: Both robots were fully capable of independent autonomous behavior and only used the executive for coordinating goal locations. To achieve this, each machine ran its own SLAM subsystem based on the Hector SLAM package[16]. The SLAM system only maintains a 2-D map for determining the x, y, θ position and heading of the robot. A 2-D SLAM system was used as it provided adequate positional accuracy with significantly lower computational load compared to a full 3-D system. For the UAV, there was an independent system that used the vertically oriented panning lidar to determine the ground plane. Since the UAV was initially mounted on top of the UGV, and during operation it was allowed to fly over obstacles, the height estimation system could not update the absolute height with every received lidar scan. On initialization, the height estimator would analyze several scans in order to determine its initial height. Then, while flying, it would filter scan points that deviated more than expected from the current estimate in order to maintain an accurate height and allow operation over obstacles. To backup this system, there was an emergency system that would automatically reset the height estimate if the perceived height exceeded a threshold even if the estimate did not. Knowing that we were operating indoors, this system ensured that we did not inadvertently collide with the ceiling.

3) Navigation: Each robot was responsible for its own navigation to the provided goal positions. We used a 3-D state lattice-based planner [17] running AD* to generate kinodynamically feasible trajectories for the UAV and a simple 2-D planner for the UGV. Upon receiving a goal, the on-board planner would perform a search on the local obstacle map to generate a trajectory from the current reported position (from the SLAM subsystem) to the goal state. In the event the robot was unable to generate a feasible trajectory (for example, if the goal was too close to an obstacle) the system would time-out and receive an updated goal from the exploration planner. The states used for planning are tuples consisting of a discretized translation in two or three dimensions, and a rotation in one dimension. The UGV planned using planar spatial coordinates and heading, $\langle x, y, \theta \rangle$, with 10cm cells[2] and 16 discretized headings, while the UAV planned in $\langle x, y, z, \theta \rangle$ with 5cm cells and 16 discretized headings. The UAV was provided with a user defined "nominal height" that it would preferentially fly at during transits, but would deviate from as necessary. In addition, the exploration planner was provided this same height and would preferentially select goal locations at this altitude. This setup allowed the planner to construct full 3-D trajectories going over, below, or around obstacles while maintaining a preferred height for the UAV to operate at.

Once the robot had a feasible trajectory, it would use the local controller to generate motor inputs to follow the trajectory. For the UGV this was performed using a trajectory rollout scheme that estimated different motions based on a small finite set of short-time horizon control inputs and selected the input that provided an endpoint closest in position and orientation to the desired trajectory. The UAV controller used a PID control for position to generate its control inputs based on the measured error between its current location and the next way-point along the desired trajectory. Altitude and yaw were handled by separate PID controllers in a similar fashion.

4) Object Detection: Besides the motion control subsystems, each robot performed its own analysis of the video feed in an effort to detect the OOI. This subsystem was based around an existing vision detection system CMVision [18]. This system was trained to detect a specific colored object; for our experiments it was a green tablet case. Like many other color based object detection systems, recalibration was required for different lighting conditions. Color-only detection was selected in order to keep the processing requirements minimized. Even so, this sub-system required the largest percentage of computing power used. When a robot detects a possible OOI, it will transmit a still image to the user (Fig. 4, right image - OOI is bottom center), pause exploration, and hold position until it receives either confirmation or rejection of the reported OOI.

C. High-Level Software

1) High Level Executive: The high-level executive is responsible for coordinating the robots. As part of this function, it provides the interface between the map merger / exploration planner module, the user interface module, and the robots.

2) User Interface: The human user only has a a few inputs to the system during run-time. The two chief inputs from the user are to start and concur with the completion of exploration. The only other input is concurrence on launching the UAV which is included for safety considerations. All

[2]In this text, we use the term "cell" to refer to 3-dimensional discretized locations defined by a center-point, $\langle x, y, z \rangle$. We use the term "state" to refer to a 3- or 4-dimensional discretized pose, $\langle x, y, (z), \theta \rangle$.

three of these inputs are handled via the User Interface module, Fig. 4. This module displays a dynamic web page to the user with zero to four buttons[3]. In addition, when one of the robots has a possible OOI detection, the best image of the OOI is forwarded to the user for confirmation. If the user confirms the OOI, the executive directs both robots to cease exploration (and presumably return to a home location and land, as appropriate). If the user denies the OOI detection, whether it is due to a false positive or other reason, the executive will direct the detecting robot, who had paused, to resume exploration. This setup reduces the cognitive load of the human operator requiring them only to judge whether the provided image is indeed of the OOI.

The interface can be accessed with any HTML browser on any device with a WiFi connection. During our testing we verified that a tablet, an Android smart-phone, and a laptop were all able to provide the high-level commands and receive the images, satisfactorily.

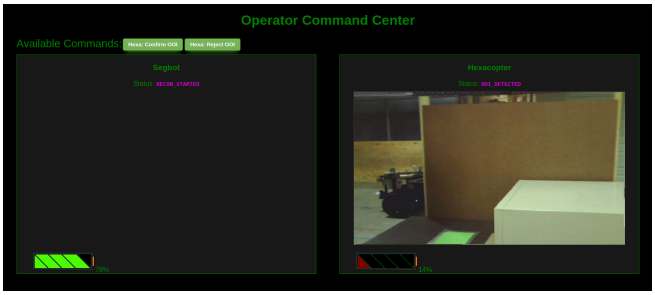


Fig. 4: The user interface. On the right, the UAV has detected an OOI and sent the image to the operator for confirmation (green tablet bottom center). Two buttons are visible along the top in green.

3) Map Merging: The map merging algorithm seeks to generate a unified 3-dimensional map of the environment by taking into account the offset between the two robots starting position. The global map was tracked in three spatial dimensions, $\langle x, y, z \rangle$, with 10cm cells and was stored as an occupancy grid [19] using an Octomap [20]. When the system initially starts, it attempts to align the two existing maps. It starts with a rough estimate of the offset between the two robots, and iterates through a finely discretized set of points around the initial estimate. This process checks for deviations in translation, $\langle x, y, z \rangle$, as well as angular deviations in heading, $\langle \theta \rangle$ (it assumes no roll or pitch errors, $\langle \phi, \psi \rangle$). Once the initial transform between the two maps is determined, this value is no longer modified. This does have the potential to cause drift errors over longer runs as the on-board SLAM system accumulates errors, however, in our testing, both robots had insignificant deviations from the global map at the time they found the OOI. Approaches such as [21] may alleviate this accumulated SLAM error if it becomes substantial.

During run-time, the map merger module uses the fixed starting transform to place new data into the global map. This data is characterized as either obstacle or free space and is represented as an occupancy grid where we store the log likelihood that the cell is either free or occupied with the middle value indicating unknown. In addition we track which cells have been viewed by the camera. To accurately annotate which cells have been seen by the camera, we must first determine which ones are free. The panning scanning laser rangefinder generates information about a wedge shape projecting from the center-line of the robot in its direction of motion. By ray-casting out towards each laser scan echo, we can identify free space between the robot and the nearest obstacle. In conjunction, with the receipt of each image, we ray-cast from the location of the camera in the direction of each camera pixel up until we reach either an unknown or an obstacle cell, or we reach the maximum effective detection range of the OOI subsystem. The set of cells traversed by the camera ray are marked as visually cleared. In this way we can positively track which cells are guaranteed to not contain the OOI in 3-D space (see Fig. 5).

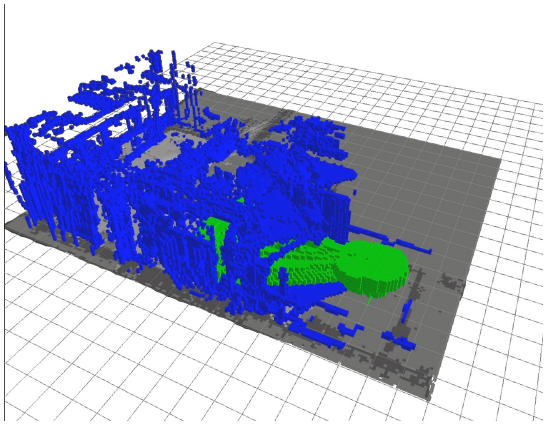


Fig. 5: Combined map. Blue objects are obstacles, green objects are cells that have been visually cleared. Free and Unknown cells are not shown.

4) Exploration: The exploration module is responsible for analyzing the environment and determining where to send each of the robots next. This module is capable of adding or removing robots from the assignment list during each iteration, if necessary, to accommodate dynamic teams or, in our case, the launching and landing of a UAV. This module can assign goal locations at any point in 3-D space to position a robot to view a particular location taking all known obstacles into account. In addition, this module is capable of differentiating which cells are visible from ground versus aerial robots and preferentially assigning those cells to the respective robot types. Technical details of the exploration planner are found in Section IV. By using the ability to determine the existence of cells not capable of being sensed by the UGV the exploration algorithm could be used to determine when to launch the UAV. This determination could be based on a large number of factors including number of these un-sensable cells, the size or arrangement of these cells, or other parameters.

[3]Available buttons are: accept OOI, reject OOI, and commence exploration, for two robots plus enter area for the UGV, of which a maximum of four are ever available at a given time.

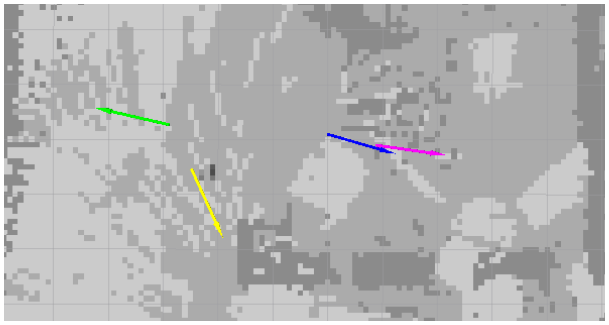


Fig. 6: Overhead view of exploration planner goals. Dark Blue is current position of the UGV, Magenta is UGV goal, Green is UAV position, and Yellow is UAV goal. The UAV has been tasked to explore the top of the desk (dark object bottom center) while the UGV is exploring a corner that has not been explored yet. In this 2-D projection, the lighter areas indicate less uncertainty in the vertical column at that point.

IV. Planning for Multi-robot 3-D Exploration

The exploration module provides goal locations for the robots to survey in an effort to minimize search time and overlap between robots. While our experiments only used a two-robot pair, the algorithm is extensible to n independent robots and has been used in past work with up to eight robots. The planner uses the combined map from the map merger module along with the points annotated as visually cleared to determine where to send each robot next. The algorithm is shown in Algorithm 1. Of note, the subscript following variable names indicates the dimensionality of the data where 3 indicates $\langle x, y, z\rangle$ and 4 indicates $\langle x, y, z, \theta\rangle$.

Algorithm 1 $G_4[\cdot]$ = GetGoal(poses $p_4[\cdot]$, robots $r[\cdot]$)

```
1:  Global: map_3, NumRobots
2:  for all i in NumRobots do
3:     CountMap_4[·] = 0
4:     InflatedMap_4[·] = InflateMap(map_3, r[i])
5:     CostMap_4[·] = Dijkstra(InflatedMap_4, p[i])
6:     FrontierPts_3[·] = FindFrontier(map_3, r[i])
7:     for all fp_3 in FrontierPts_3[·] do
8:        ViewPts_4[·] = Views(FrontierPts_3[fp], r[i])
9:        for all vp_4 in ViewPts_4[·] do
10:          CountMap_4[vp_4]++
11:       end for
12:    end for
13:    for all pt_4 in CostMap_4[·] do
14:       score_4[p] = ScorePt(CountMap_3[pt_4],
15:                            CostMap_4[pt], G)
16:    end for
17:    G_4[i] = argmax(score_4)
18: end for
19: return G_4[·]
```

The inputs to the GetGoal function are the $\langle x, y, z, \theta\rangle$ poses p of all of the robots and a parameter list r providing the footprint, nominal altitude ($= 0$ for UGV), sensor field of view, and sensor position and orientation relative to the robot body frame, for each robot.

During each planning iteration the exploration planner will generate new goals to all robots (that are ready to accept new goals) based on the current map. In practice, we transmit a goal to each robot as soon as the goal is determined and repeat the loop continuously.

The planner first inflates the obstacles in the combined map to account for the footprint of the robots (line 4). Because of having possibly non-circular robots, the inflated map has the robot heading θ as a dimension. The inflation is followed by a Dijkstra Search starting from the current location of the robot to determine the cost to each accessible state in the environment (line 5). Once these two initial processing steps are completed, the map is parsed to determine all of the frontier cells. A standard definition of a frontier cell is an unknown cell that is directly adjacent[4] to a known free cell. We use a slightly modified definition of this in that we consider a visually cleared cell as known, and all others, even if we have laser data on that cell, as unknown (line 6). By using this approach, we can guarantee that the robot will find the OOI, given sufficient battery and permissible obstacle configuration[5], as it will maneuver to view all visually unknown cells within the environment.

Since our goal is to determine information about the map, selecting goal points with higher information gain is beneficial. We approximate the total information gain accrued by traversing to a given 4-dimensional state by the number of frontier cells visible from that state. To accomplish this, for each frontier cell we determine the 4-dimensional set of states that the robot could be at in order to successfully visually clear the frontier cell. This requires knowledge of the robot sensing model including the field of view of the camera system and its mounting location as well as the maximum range the OOI detection algorithm can robustly detect the object at (lines 8-11). The sensing model was used to ray-trace within the field of view of the sensor from the robot out to the nearest obstacle and to subsequently annotate the intervening states as obstacle-free.

Finally, each potential state in the reachable space receives a score based on how many frontier cells are visible from that state, the distance the potential goal state is from the current state, and a penalty term. The distance and count terms are weighted by a user-defined parameter, $0 \leq \xi \leq 1$ that adjusts the propensity to move farther to get a higher information gain or to select a nearby but not very lucrative state (1). For $\xi = 1$ the planner will select the state with the highest information gain, ignoring the distance term. Conversely, setting $\xi = 0$ will result in the planner selecting the lowest cost state without regard to the information to be

[4]For our system we define "directly adjacent" to mean cells that differ along only a single dimension by one unit, i.e. $\langle x, y, z\rangle$ and $\langle x, y+1, z\rangle$ are adjacent but $\langle x, y, z\rangle$ and $\langle x+1, y+1, z\rangle$ are not. Note: frontier cells are not defined by any heading information.

[5]We assume that a valid configuration contains a state accessible from the starting state and from which the OOI may be sensed.

gained.

$$score[i] = \frac{count[i]^{\xi}}{cost[i]^{(1-\xi)}} \cdot penalty[i] \tag{1}$$

The penalty term can incorporate a wide range of user preferred behavior. For our system, this term was constructed to downgrade states that are in close proximity to any other robots goal state. In addition, this term also penalized very short range motions that are harder to execute (2). The threshold_L and threshold_D values are set by the user[6].

$$penalty[i] = \textsc{Length}(i) \cdot \min_{j, i \neq j} \Big(\textsc{Proximity}\big(i, G(j)\big) \Big) \tag{2}$$

$$\textsc{Length}(a) = \begin{cases} 1 & \textsc{CostMap}(a) \geq \text{threshold}_L \\ \frac{\textsc{CostMap}(a)}{\text{threshold}_L} & \text{otherwise} \end{cases}$$

$$\textsc{Proximity}(a, b) = \begin{cases} 1 & \text{dist}(a, b) \geq \text{threshold}_D \\ \frac{\text{dist}(a,b)}{\text{threshold}_D} & \text{otherwise} \end{cases}$$

Due to the UAV's limited flight time, we provide a further enhancement to maximize the UAV's value. When determining the frontier points on line 6, the exploration planner first considers only those points that are not visible to the UGV. In this way, the planner will first send the UAV to cover portions of the environment the UGV cannot sense. Once it has exhausted the UAV-only points without finding any candidates, it will reevaluate based on all frontier points[7].

In order to determine which states are or are not visible to the UGV, we ray-cast from the potential target state to the set of states that the UGV sensor could be located in (taking into account orientation and obstacles). If all rays encounter obstacles then the state is not visible. In all other cases there exists at least one viable configuration of the UGV that will allow the target state to be sensed.

V. Experiments

A. Setup

Our experiments aimed to validate our entire approach to exploration by having the robot team search a previously unknown area attempting to locate a particular tablet (our OOI) identified by its unique color. The operator stood outside the area and issued the allowed commands with no other interaction with the robots. We defined a successful run if either of the robots were able to identify the tablet before the UAV depleted its battery. Depending on the length of time spent airborne, and to a lesser extent the amount of processing load and time spent operating while mounted to the UGV, the UAV is limited to between four and ten minutes of flight, compared to several hours of exploration time for the UGV. An unsuccessful run was one in which after conducting a search the robots were not able to find the tablet before a low battery forced the UAV to land[8].

We conducted our experiments in an enclosed indoor area measuring approximately 30m $\times$ 10m $\times$ 5m of which the upper 2-3m were occupied with pipes and conduit. The area was partitioned into sections with movable walls and objects such as a desk and filing cabinet were placed inside the area. This resulted in approximately 400,000 cells capable of being detected and analyzed, depending on obstacle placement and room configuration. The robots started from the same location for all tests near the edge of the search area and had a predetermined first goal located 5m into the exploration zone that the UGV moved to when commanded to "enter the area".

We had different people place the OOI during our experiments to rule out any bias in selecting locations that were particularly easy or hard for the system. The direction provided to the person placing the OOI was to place it so that the OOI would only be visible to the UAV once it was airborne. This was done by placing the OOI in a location not visible from the UAV prior to it taking off from the UGV and not visible to the UGV. Separate tests were performed to verify the UGV was capable of also detecting the OOI, and it was successful on all runs.

The test environment had one object that had an identical color as the OOI and served as a decoy for the detection system by causing a false positive. In the event the decoy was detected, the operator rejected the classification and resumed the exploration. Decoy detection was not deemed to be a failure but the time spent pausing and waiting for the operator to reject the OOI was counted towards the completion time.

Since the exploration algorithm has several user selectable parameters, we set them as follows: To achieve balance between distance traveled and information gain, we set $\xi = 0.5$, to facilitate the robots spreading out, we set $\text{threshold}_D = 5$, and to discourage very short range goals, we set $\text{threshold}_L = 1.2$ for all runs. In addition, while the map merger process can determine when to launch the UAV based on detecting unreachable frontier states, for all of the test runs, we had the UAV launch at the first opportunity after 2 minutes 30 seconds of UGV only exploration[9]. The 2:30 takeoff time was based on prior experiments that indicated a reasonable percentage of the UGV accessible space was explored by that point on average (for an example, see Fig. 7b for the case where the UAV took off late - the UGV exploration progress plateaus at approximately 2:30).

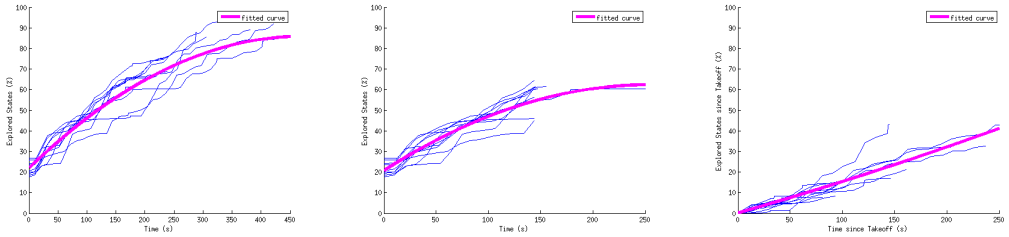
The cameras used on both robots were identical Logitech C310 webcams that provided images at 1280 $\times$ 720 pixels at 25Hz. The UGV camera was mounted approximately 20cm

[6]The Length threshold_L duplicates to some extent the ξ parameter. The difference being that threshold_L is an absolute value - the penalty is applied independent of the information gain - while the ξ parameter only changes the relative importance of cost vs. information gain.

[7]This process may transition multiple times between UAV-only and all frontier points as the environment is explored and new obstacles discovered.

[8]The action was automatic and occurred when battery voltage under load averaged less than 13.8V for 10 seconds. Runs terminated due to a mechanical failure of a robot were not counted in the results.

[9]There was one area of the environment that the overhead obstacles were too low to allow for the UAV to launch which caused the takeoff to be delayed on two occasions - once by 10 seconds (run 5) and once by 2:25 (run 1).



(a) Full Run (b) Prior to Takeoff (c) After Takeoff

Fig. 7: Percentage of cells explored over time (based on full volume of search area). Heavy magenta line indicates best fit.

from the ground along the vehicle center-line with the camera axis parallel to the ground. The UAV camera was mounted on the UAV forward axis and tilted downward by 10° from horizontal. With the UAV nominal flying height set to 1.65m, the camera typically was at 1.6m.

B. Results

TABLE I: Experimental Results.

Run	Detect OOI	Detect Decoy	% of Env. Explored	Time to Detect OOI	Time Since UAV Launch
1	✓	✘	55.6	7:08	2:23
2	✓	✓	63.3	5:51	3:21
3	✓	✓	57.9	4:49	2:19
4	✓	✘	50.5	4:01	1:31
5	✓	✘	45.6	3:48	1:08
6	✓	✘	49.3	3:50	1:21
7	✓	✘	45.3	3:56	1:26
8	✘	✘	60.5	7:00	4:30
9	✓	✘	58.5	6:20	3:50
10	✓	✘	56.2	5:02	2:32

We made 10 complete runs with the results as shown in Table I. Only 1 run failed to find the OOI within the time allowed due to the UGV exploring the particular corner containing the OOI (but at too low an altitude) prior to the UAV launch. Since the area was already predominately explored, the UAV did not return to verify the small remaining unknown area during the available time. Two runs detected the decoy initially, but after the operator rejected the false positive, they both successfully found the actual OOI.

Using 400,000 as an approximation of the number of detectable cells[10] in the environment we can estimate the percentage of the environment covered prior to discovery of the OOI, Fig. 7. For all of the runs we cover 80-90% of the environment prior to detection of which 40-60% is discovered by the UGV prior to the UAV taking off. As can be seen in Fig. 7b, the UGV progress has already begun to plateau by this point. The UAV continues to make progress as it explores areas inaccessible to the UGV accounting for an additional 10-30% before detecting the OOI or landing[11], Fig. 7c.

Since the UAV preferentially visits locations that the UGV is incapable of exploring, it detected the OOI within the first few goals (1:30 of flight) almost half of the time.

[10]The remainder being internal to obstacles.

[11]The values in Fig. 7c are a combination of both UAV and UGV information, however, since we know that there are significant portions of the map that are inaccessible to the UGV, and the pre-launch data is plateauing, we surmise that the majority of the gain post-launch if from the UAV.

VI. Conclusion & Future Work

Our contribution is two-fold: we have introduced an air-ground robotic system capable of operating autonomously and an algorithm for performing exploration in cluttered 3-D environments. With our experiments, we have shown how planning for heterogeneous teams of robots can be an effective method of exploring environments that either robot alone would be unable to explore. Our results show that the system is capable of entering, exploring, and detecting an OOI within the time constraints imposed by the limited battery life of a UAV. In the future, we will expand the system to include robots with different sensor packages in addition to the different modes of locomotion shown in this work as well as examining how to optimize the threshold used for launching the UAV. In addition, we will look to making the high-level executive a distributed process that can remain functioning with the loss of any one robot.

References

[1] B. Yamauchi, "A frontier-based approach for autonomous exploration," in *Computational Intelligence in Robotics and Automation, 1997. CIRA'97., Proceedings., 1997 IEEE International Symposium on*, July 1997, pp. 146 –151.

[2] ——, "Frontier-based exploration using multiple robots," in *Proceedings of the second international conference on Autonomous agents*, ser. AGENTS '98. New York, NY, USA: ACM, 1998, pp. 47–53. [Online]. Available: http://doi.acm.org/10.1145/280765.280773

[3] R. G. Simmons, D. Apfelbaum, W. Burgard, D. Fox, M. Moors, S. Thrun, and H. L. S. Younes, "Coordination for multi-robot exploration and mapping," in *Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on Innovative Applications of Artificial Intelligence*. AAAI Press, 2000, pp. 852–858. [Online]. Available: http://portal.acm.org/citation.cfm?id=647288.723404

[4] A. Visser and B. Slamet, "Balancing the information gain against the movement cost for multi-robot frontier exploration," in *European Robotics Symposium 2008*, ser. Springer Tracts in Advanced Robotics, H. Bruyninckx, L. Preucil, and M. Kulich, Eds. Springer Berlin / Heidelberg, 2008, vol. 44, pp. 43–52.

[5] R. Zlot, A. Stentz, M. Dias, and S. Thayer, "Multi-robot exploration controlled by a market economy," in *Robotics and Automation, 2002. IEEE International Conference on*, vol. 3, 2002, pp. 3016 –3023.

[6] W. Burgard, M. Moors, C. Stachniss, and F. E. Schneider, "Coordinated multi-robot exploration," *Robotics, IEEE Transactions on*, vol. 21, no. 3, pp. 376–386, 2005.

[7] J. Butzke, K. Daniilidis, A. Kushleyev, D. D. Lee, M. Likhachev, C. Phillips, and M. Phillips, "The university of pennsylvania magic 2010 multi-robot unmanned vehicle system," *Journal of Field Robotics*, vol. 29, no. 5, pp. 745–761, 2012. [Online]. Available: http://dx.doi.org/10.1002/rob.21437

[8] J. Butzke and M. Likhachev, "Planning for multi-robot exploration with multiple objective utility functions," in *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*. IEEE, 2011, pp. 3254–3259.

[9] P. Sujit and R. Beard, "Multiple uav exploration of an unknown region," *Annals of Mathematics and Artificial Intelligence*, vol. 52, no. 2-4, pp. 335–366, 2008. [Online]. Available: http://dx.doi.org/10.1007/s10472-009-9128-7

[10] R. Sawhney, K. M. Krishna, and K. Srinathan, "On fast exploration in 2d and 3d terrains with multiple robots," in *Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems - Volume 1*, ser. AAMAS '09. Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems, 2009, pp. 73–80. [Online]. Available: http://dl.acm.org/citation.cfm?id=1558013.1558022

[11] K. Yang, S. Keat Gan, and S. Sukkarieh, "A gaussian process-based rrt planner for the exploration of an unknown and cluttered environment with a uav," *Advanced Robotics*, vol. 27, no. 6, pp. 431–443, 2013. [Online]. Available: http://dx.doi.org/10.1080/01691864.2013.756386

[12] C. Rasche, C. Stern, L. Kleinjohann, and B. Kleinjohann, "A distributed multi-uav path planning approach for 3d environments," in *Automation, Robotics and Applications (ICARA), 2011 5th International Conference on*, Dec 2011, pp. 7–12.

[13] R. Vidal, O. Shakernia, H. J. Kim, D. H. Shim, and S. Sastry, "Probabilistic pursuit-evasion games: theory, implementation, and experimental evaluation," *Robotics and Automation, IEEE Transactions on*, vol. 18, no. 5, pp. 662–669, 2002.

[14] H. Tanner, "Switched uav-ugv cooperation scheme for target detection," in *Robotics and Automation, 2007 IEEE International Conference on*, April 2007, pp. 3457–3462.

[15] B. Grocholsky, J. Keller, V. Kumar, and G. Pappas, "Cooperative air and ground surveillance," *Robotics Automation Magazine, IEEE*, vol. 13, no. 3, pp. 16–25, Sept 2006.

[16] S. Kohlbrecher, J. Meyer, T. Graber, K. Petersen, U. Klingauf, and O. von Stryk, "Hector open source modules for autonomous mapping and navigation with rescue robots," in *RoboCup 2013: Robot World Cup XVII*. Springer, 2014, pp. 624–631.

[17] B. MacAllister, J. Butzke, A. Kushleyev, H. Pandey, and M. Likhachev, "Path planning for non-circular micro aerial vehicles in constrained environments," in *Robotics and Automation (ICRA), 2013 IEEE International Conference on*. IEEE, 2013, pp. 3933–3940.

[18] J. Bruce, T. Balch, and M. Veloso, "Fast and inexpensive color image segmentation for interactive robots," in *Intelligent Robots and Systems, 2000. (IROS 2000). Proceedings. 2000 IEEE/RSJ International Conference on*, vol. 3, 2000, pp. 2061–2066 vol.3.

[19] A. Elfes, "Sonar-based real-world mapping and navigation," *Robotics and Automation, IEEE Journal of*, vol. 3, no. 3, pp. 249–265, June 1987.

[20] A. Hornung, K. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "Octomap: an efficient probabilistic 3d mapping framework based on octrees," *Autonomous Robots*, vol. 34, no. 3, pp. 189–206, 2013. [Online]. Available: http://dx.doi.org/10.1007/s10514-012-9321-0

[21] J. Jessup, S. Givigi, and A. Beaulieu, "Robust and efficient multi-robot 3d mapping with octree based occupancy grids," in *Systems, Man and Cybernetics (SMC), 2014 IEEE International Conference on*, Oct 2014, pp. 3996–4001.