Guidance, Navigation, and Control and Co-located Conferences
August 19-22, 2013, Boston, MA
AIAA Guidance, Navigation, and Control (GNC) Conference

AIAA 2013-4989

Indoor GPS-denied Context Based SLAM Aided Guidance for Autonomous Unmanned Aerial Systems

Dmitry Bershadsky[*] and Eric Johnson[†]
Georgia Institute of Technology, Atlanta, GA 30332-0150, USA

Autonomous exploration and mapping of environments is an important problem in robotics. Efficient exploration of structured environments requires that the robot utilize region-specific exploration strategies and coordinate with search other agents. This paper details the exploration and guidance system of a multi-quadrotor unmanned aerial system (UAS) capable of exploring cluttered indoor areas without relying on any external aides. Specifically, a graph-based frontier search algorithm which is aided by an onboard Simultaneous Localization and Mapping (SLAM) system is developed and flight tested. A technique is developed in for segmenting an indoor office-like environment into regions and to utilize the SLAM map to conduct specific activities in these regions. A goal-directed exploration strategy is created building on existing hybrid deliberative-reactive approaches to exploration. An obstacle avoidance and guidance system is implemented to ensure that the vehicle explores maximum indoor area while avoiding obstacles. The environment is explored and regions are segmented by detecting rooms and hallways which expedites the search. The multi-vehicle system is Georgia Tech Aerial Robotic Team's entry for the annual International Aerial Robotics Competition (IARC).

I. Introduction

If robots are intended to co-exist with and be useful to humans in their environments, it is important for those robots to understand how to explore and navigate within them. Robots must also effectively navigate and move around within these areas without a priori knowledge of the area. Many guidance systems rely on reactive only algorithms, such as obstacle avoidance or wall following. However, this method has well documented problems [1] such as local minima and inefficient and sometimes incomplete exploration. To alleviate this, Yamauchi [2] invented the exploration strategy based on frontiers. Graph based systems with frontier following have also been studied with increased exploration coverage. Moorehead [3] used frontiers that are created utilizing multiple sources of information. Guiding the vehicle based on an undirected graph allows the system to depart from previous version of the

*Graduate Research Assistant, School of Aerospace Engineering, dbershadsky@gatech.edu.
†Associate Professor, School of Aerospace Engineering, eric.johnson@ae.gatech.edu.

system, which is tree based, such as that described by Julia et.al. [1]. This type of guidance increases time efficiency of search. Graphs allow for the closure of loops whereas a tree does not allow this, because a tree has a parent-child format and does not allow for child-child connections. Although the tree method is well structured and relatively computationally non-intensive, it does not provide the most optimal path planning in terms of time and distance to the next waypoint. That is, if the current waypoint has no new frontiers (children), the vehicle must return to the parent of a child before exploring the next child. The inefficiency becomes apparent when the parent is located further from the current position than the next unexplored child. This means that should a vehicle using a tree based graph come upon another branch and find no further frontiers, it would need to backtrack down the branch that it is on to continue down the second branch. A graph allows branch-branch connections to be made and saves backtracking time.

There has been a body of work focused on extracting the topological and semantic information of the explored environment using both geometric and other extracted features. Kuipers [4] introduced the Spatial Semantic Hierarchy, a model for representing large-scale space and utilized multiple-interacting representations such as region-specific control inputs, and geometric characteristics to form a topological representation of the environment. Kuipers [5] utilized images labeled with its proximity to other images in sensory space to learn the similarity of different images. This learned information as well as the topological maps and views are used to cluster different regions from an explored environment. Konolige [6] utilized images purely as features for building topological maps. Semantic information about a region like a hallway has been utilized by Stachniss [7] to expedite multi-robot exploration by providing a higher incentive for robots to stay in hallways and hence preventing multiple robots from entering rooms, and thereby becoming bogged down and missing goals further down the hallway for some time. Dellaert and Bruemmer [8] and Oberlander [9] proposed SLAM algorithm based on FastSLAM that maps features representing regions with semantic type, topological properties and an approximate geometric extent. Detection of rooms to aide the guidance system is discussed, as well mention of using visual cues, such as directional signs [12]. Partioning of maps such as this have been addressed by Ekvall [10], where he applied an automatic strategy for map partitioning based on detecting borders between rooms and narrow openings. Human-augmented map partitioning has been addressed by Nieto [11] where semantic regions are expressed as a mixture of Gaussian distributions and human inputs to obtain labeling for each region. This has been done with the intent of reaching different areas in the map after the areas have been explored. This work relates closely with those presented in [13], [14], in that they utilize robots with limited computing resources.

Goal directed search, for example searching for a specific item such as a USB drive in an office environment, is a specific type of search, and the type focused on by this paper. If it is known that the

USB drive is likely to exist in a cubicle, the situation then requires that the robot first identify cubicles, as they have higher probabilities of containing a USB drive. It is efficient to search for the USB drive more carefully there, than for example, in hallways. This is the premise of the 6th mission of the IARC; that is, a USB drive is located in a simulated office environment. The vehicle will have little a priori information other than there is a drive located inside a security office. Signs on the walls help guide the vehicle inside the arena; their use by the guidance system is described but their detection is outside the scope of this paper. The system developed performed in the 2011 and 2012 IARC events in North Dakota. Flight testing of the guidance system is discussed in the results section.

In this paper, we introduce an approach to the utilization of the SLAM data to aide the exploration of the environment. In terms of processing power, a practical, cooperative search guidance system is described and flight tested onboard a UAS with limited power and processing recourses. Four main guidance mechanisms are assisted by SLAM data: frontier creation, graph pruning, en route obstacle and local minimum detection, and room and hallway classification. Unlike with other methods, with the guidance algorithm described here, frontiers are chosen using the available SLAM data to keep the vehicle out of areas that it has already cleared. In addition, the graph is built and updated constantly based on the SLAM data such that paths through known free space may be chosen. An algorithm is also introduced to aide the onboard classification of rooms and hallways inside an office-like environment. This classification along with the other three methodologies expedite the search of the environment by considering the context of the vehicle's state with respect to the goal of the search. The regions are classified by studying the SLAM map's features during exploration which provides context to the guidance system, allowing it to make time-saving choices during execution. A modified CoreSLAM algorithm is used for this UAS. Guidance-relevant modifications are described below. The system is scalable by means of introducing multiple, communicating search agents into the environment.

II. Flight Platform and Avionics

The Georgia Tech Quadrotor (GTQ) is an AscTec Pelican quadrotor, the platform utilized to explore the environment in the IARC and to execute the guidance software described in this paper.

Hardware. This fixed-pitch quadrotor is a standard platform in the UAS research community. The vehicle carries a custom avionics suite including two Overo Fire ARM Cortex-A8 OMAP3530 processor boards with Tobi expansion boards, a custom ATmega 128 microcontroller SAS with an Analog Devices ADIS-16365-BLMZ IMU, a Hokuyo URG-04LX-UG01 2D LIDAR, an Maxbotics LV-MaxSonar-EZ1 sonar, and a Point Grey Research Inc. Firefly MV camera.

Software. A Simultaneous Localization and Mapping (SLAM) algorithm is used to fuse information from the LIDAR sensor, the inertial measurement unit (IMU), and sonar to provide relative position, velocity, and attitude information. The SLAM system is implemented by Sobers et.al.[14]. The

CoreSLAM algorithm is modified to not clear pixels of obstacles where no returns are detected. This cleans up the SLAM data by discounting erroneous regions where the critical angle of the laser is exceeded. The effect is clearly seen by comparing the SLAM map in figure 3 right to that in figure 7. The SLAM information is then used to direct the vehicle's motion as it searches an unknown environment via the guidance system detailed by this paper.

III. Guidance System

Exploration Guidance System Flow. The guidance system is nestled under the obstacle avoidance system onboard the vehicle. As input, the guidance system takes mainly data from the Simultaneous Localization and Mapping (SLAM) algorithm. In addition to LIDAR data of the current scan of obstacles around the vehicle, the SLAM system provides positioning information in the local SLAM frame of the physical environment in the form of a north and east position and a heading. The SLAM coordinate frame is initialized by using north as forward and east to the right, relative to the vehicle's body coordinate system. The output of the guidance system is a north, east location which becomes the positional goal of the vehicle. The position is used by the controller to create a velocity command that is fused with that created by the obstacle avoidance system. The resultant command drives the vehicle in the lateral north-east plane.

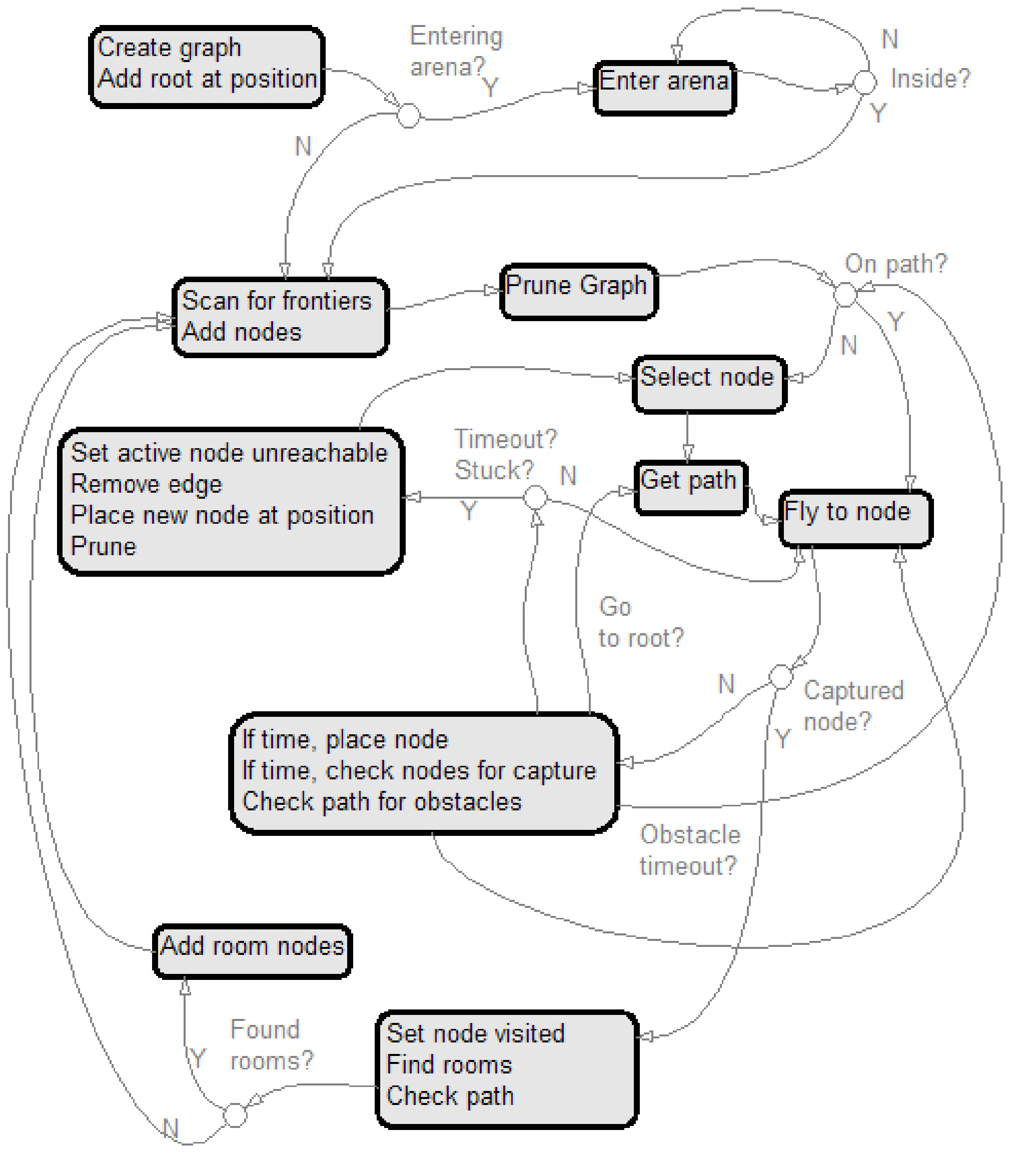

Figure 1. Guidance system state flow

Initialization. Figure 1 demonstrates the high-level flow of the guidance system. The exploration system initializes by creating a graph and adding a root node at it's SLAM position. This is the node to which the vehicle returns should it be commanded to go back to the entry point for a completed or aborted mission.

Frontiers and Nodes. The developed guidance system uses available information gathered through the onboard sensors, which include a LIDAR scanner. Scan frontier points are used to add nodes to the undirected exploration graph at each scan point. A frontier is a point in the laser scan at the time of processing which is either a discontinuity with respect to range of consecutive returns, or an arc of no returns. In the latter case, the frontiers at the two discontinuities are lumped into the non-return arc. Figure 2 shows a view of the simulated vehicle at the point of arena entry. The red nodes indicate frontiers at that scan.

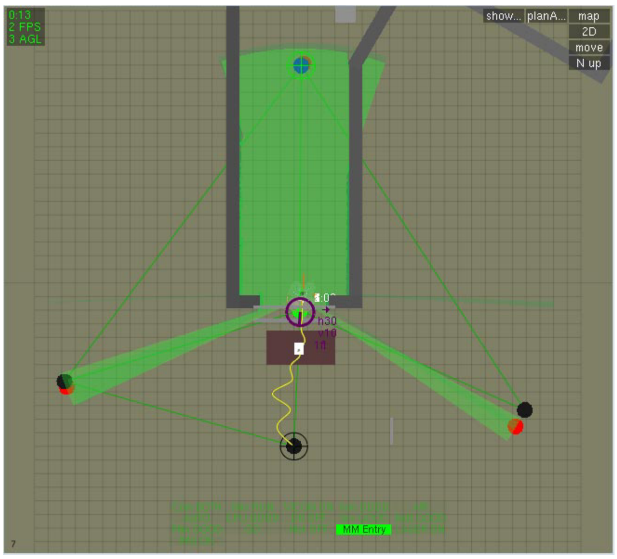

Figure 2. Top view; simulated vehicle entering hallway along yellow path, carrying a 240° LIDAR scanner (green), building exploration graph.

A frontier point becomes a node in the exploration graph if it passes certain criteria. A node is added (blue, Fig. 2, from the Georgia Tech UAV Simulation Tool - GUST) as the midpoint of each independent frontier should it be of a minimum specified arclength. This is done so that small areas of non-returns and discontinuities do not pollute the exploration graph. These are typically either areas where the laser passes through small openings or shallow incidence angle regions. In either case, travel to those frontiers is not desirable. In contrast to many existing methodologies, a frontier will not be created if the SLAM data suggests that the area has already been explored. Each new node is checked against all existing nodes. Should a node exist within a specified Cartesian distance from the new node, the new node will not be added, but a graph connection (green lines Fig. 2) will be made between the existing node and the node at the current scan location.

Nodes may be added to the graph in three situations. The first method, via frontiers, is as described above. A second method is to place nodes periodically along the path as the vehicle explores at its SLAM position. Leaving this "crumb trail" allows the vehicle to have more options should the SLAM map shift during exploration. Also, the frontier scan is activated during such a node placement. Otherwise, small openings such as doorways may be missed while en route to a distant waypoint, especially in narrow hallways. The third method, described in a later section, is via a room detector.

A node is defined in the graph by a unique ID, north and east positions, frontier size used to generate it, and a status (unvisited, visited, unreachable, off-limits, and room). A visited node is defined as one which has seen the vehicle within a specified distance of itself, and an unvisited node is classified if the opposite is true. Unreachable nodes are those that the vehicle attempts to visit but cannot. An off-limits node is one outside of the arena or in another specified off-limits area. A room node is defined as one belonging to an area classified as a room. These are described in further detail by the following sections.

Edges. Edges are undirected connections between nodes in the graph. They signify a path free of obstacles from node to node. Several properties define an edge: Cartesian length and end-point node unique IDs. Edges are added to new nodes from the location of the vehicle at the time of the scan.

Node Weighting and Path Selection. As the vehicle captures a node, a new scan is performed and the next node is chosen based on a weighting system. Distance to the node, the arclength of the frontier used to create it, node classification, and a directional bias are used to select the next positional goal. Algorithm 1 details the selection process.

Algorithm 1: Node Selection

```
If go to start, best node = 0
for each $node_i$ in the graph,
    Create matrix of lengths from captured $node_{active}$ to $node_i$
    If $node_i$ length is not finite or is 0, or is south of window, mark it as off limits
    If $node_i$ is in a room, weight *= large number
    If status = unvisited or room, unexplored nodes++
    If status = unvisited or unreachable or room, numUnexpUnrNodes++
    If status = visited or off limits, i++
    If numUnexploredNodes && status = unreachable
    weight ~= distance gain * room gain * size gain * directional gain
    distance = gain * 1/length
    room gain = large number if node is inside room
    size gain = gain * size
    direction gain = gain * node hdg o favor direction
    if (weight > best_weight)
        best_weight = weight
        best_vertex = i
    end if
end for
```

Path Selection. Once the best node is selected based on the criteria above, the guidance system then finds the most length-optimal graph edge path to it. The Dijkstra shortest path algorithm is used for this calculation. It considers at all possible paths to the chosen node with node weights and edge lengths along the way.

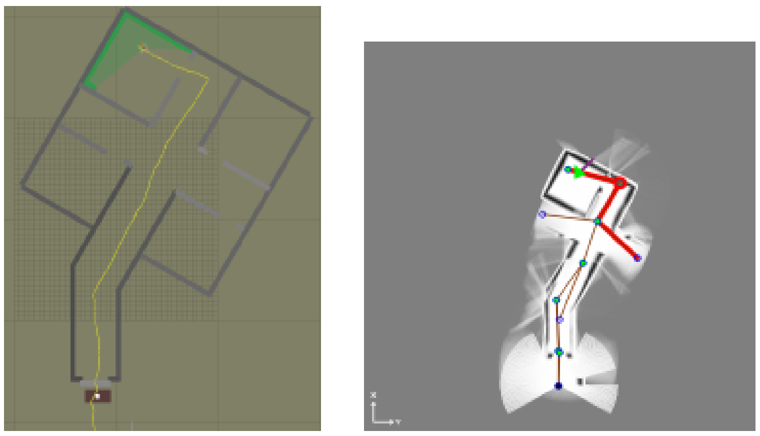
Figure 3. Graph based guidance in the simulated 2011 competition arena (left) Simulated truth map and vehicle exploration path (right) Simulated SLAM solution and exploration graph overlay

The intended graph path is highlighted as well. Simulated truth results are shown in fig. 3(left) where the yellow path traces the vehicle's position over time. The vehicle has traveled approximately 120' in the figure, showing about 60 seconds of exploration.

The vehicle continues to build the graph which can quickly become complex, as shown in figures 5, a continuation of figure 2.

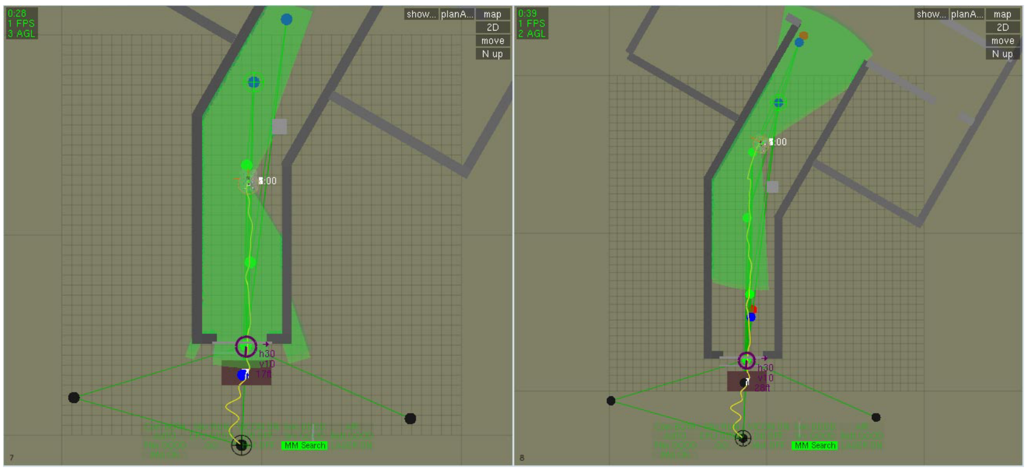

Figure 4. Continuation of graph building process (left) simulation t = 28s, (right) t = 39s

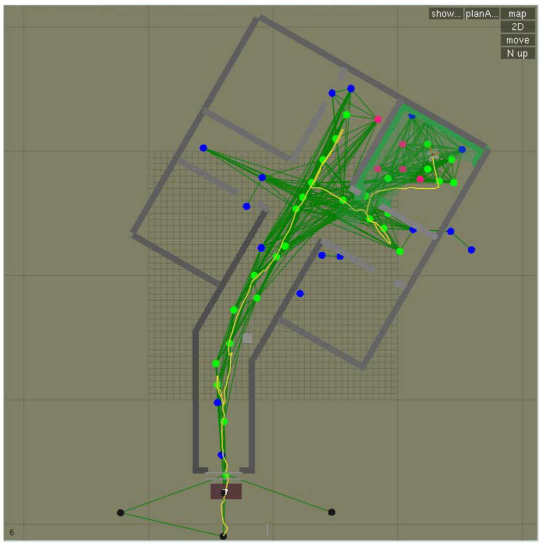

Figure 5. Exploration graph as simulated vehicle enters north-east room of the 2011 IARC.

IV. SLAM as Guidance Aide

To aide the guidance system during exploration, the SLAM map is used, in addition to data from available sensors. This dramatically reduces the search time inside the simulated arena discussed in this paper. The effect of specific changes are described in the results section.

Frontier creation. Unlike many other methods that use the current laser scan to create frontiers, the SLAM map data in this system is fused with the current scan in order to expedite the search.

Frontiers are not upgraded to nodes should they reside in a LIDAR-cleared area in the SLAM map. This is in an effort to drastically reduce unnecessary backtracking during exploration.``

Graph Pruning. The map is used to modify the exploration graph. The exploration graph is 'pruned' using the obstacle map. As there is currently no tactical path planner onboard the vehicle and nodes may be unreachable due to erroneous frontiers or SLAM map shifting, a pruning algorithm (2a) is developed to disconnect edges to these nodes.

Algorithm 2a: Graph Pruning

```
for each node_i in the graph,
    for each node_j in the graph,
        If node_i != node_j and node_i not off limits
            Start at node N, E position of node_i
            Step in direction of node_j
            Check SLAM array with specified kernel
            If obstacle
                obsCount ++
            end if
            If obsCount > maxObsCount
                Remove this edge
            end if
    end for
end for
```

A complimentary algorithm (2b) is also developed to connect all nodes with a clear obstacle map line of sight to each other.

Algorithm 2b: Graph Neighbor Adding

```
    …
            If obsCount > maxObsCount
                Do not add this edge
            end if
…
```

Room detection. Data abstraction may be beneficial should a goal directed search be driven by a small target in a expansive environment that has contextual probability, such as demonstrated by Rottmann et.al.[2]. For example, in the 2012 IARC, the USB target is known to be located in a security office in the simulated complex and expedient retrieval is necessary both due to the scenario and the limited flight time of the vehicle. It is thus beneficial to traverse hallways quickly and slow the search when inside a room so that the vehicle might pass over the target and have sufficient time to detect it. Also, it is known that the USB drive is located in a particular office. Thus, finding the sign labeling this office also potentially expedites a search. Sign detection is another method that will be described in the following sections.

Graph nodes may become distributed in such a way such that the object may not be readily detected by a downward facing camera. This can stem from infrequent scanning and placing frontiers only at the midpoint of each scan range discontinuity. Also, as the vehicle explores the environment, a radius condition is used to determine waypoint capture. The effect on a search for a small object in a large area is that the camera searching for it may not pass over it because the entire room area will not be searched. To remediate this, a room detector is developed to fill room areas with high-weight nodes.

The detector uses a SLAM map as input. The SLAM map is modified by image processing filters to attempt to find room regions. A room classifier is developed for this purpose. The algorithm first searches for enclosed regions. To find these, the obstacle map is dilated. The main purpose of this action is to close doorways to allow the algorithm to disconnect rooms from hallways. The map is then thresholded to create a binary map. This is done because the voxel values in the SLAM map are probability based and thus are not binary, which is required for contour generation. Contours are then drawn around all connected regions of free space. Each contour is then checked for sizing criteria of the room classifier. Bounding boxes are then calculated around each contour.

The rooms should be:

- of a maximum and minimum contour perimeter. Also, rooms
- of a maximum aspect ratio, length to width
- of a maximum area ratio with respect to their respective bounding box

All connected regions passing these criteria are marked as rooms. Once a room is found, the algorithm scatters nodes around the room in a grid pattern of specified spacing and dimensions. The pattern may be selected as having fixed extents, or may grow radially from the center of pixel mass of each room. The free path algorithm described above is used to bound the growth of the latter to the confines of the room, shown in figure 6.

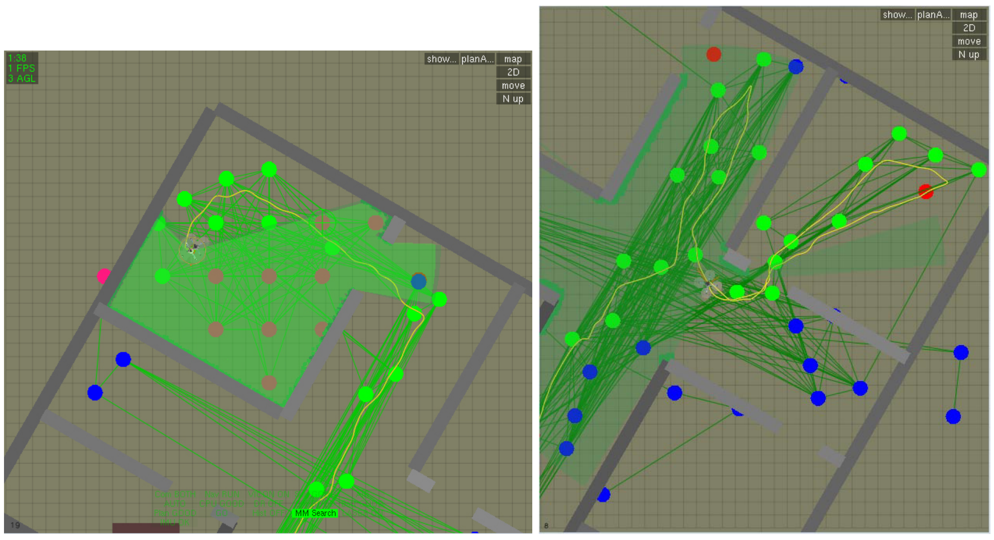

Figure 6. Room node placing algorithm – pink nodes indicate 'room' status. Room detector on (left) and off (right)

This allows the room to thoroughly be searched. The contour bounding box is also used to keep the vehicle from re-entering rooms that have already been cleared. This increases the efficiency of the search with respect to time spent. Results of this system are shown in the results section.

En Route Obstruction Detection. Because of the limitations described above of purely following edges, the vehicle may be stuck in unfavorable locations in the environment and not be able to re-access the exploration graph. The guidance algorithm constantly checks for any obstructions in the path of the vehicle to lessen this possibility. The SLAM map is used here to help the vehicle while en route to a node. Bresenham's line algorithm is used to check for obstacles in the SLAM map. The free path algorithm constantly checks the path from the vehicle to the destination node for obstacles. A timeout is triggered should an obstacle be detected for more than a specified amount of time and the vehicle has a low velocity; that is, it's stuck in a local obstacle minimum.

In addition, should the vehicle not reach the target node in a specified amount of time, or should the software detect that the vehicle is stuck in an unfavorable position, the vehicle will place a node, connect it to the graph via the pruning algorithm, and select a new node. Stuck detection works by applying three criteria, which all increase the confidence in the stuck detector output. Firstly, the mean velocity of the vehicle over a rolling time window is calculated and checked against a threshold set in the software. If the velocity is below this value, the probability of a stuck detection is increased. Secondly, in that same time window, should the velocity of the vehicle flip directions on itself more than a number of allowable times, the probability is yet again increased. Lastly, if the vehicle does not leave a circle of a specified radius within the time window but has traveled a specified distance, the stuck probability is again increased. Timeout is forced should probability reach a specified value.

V. Multi-Agent Search

Multiple vehicles may also be used in order to coordinate a more efficient search. Especially in a large environment, to an extent, [1], [2], [3], and [7] show that the more agents the more time-efficient the search. Decentralized coordination between multiple vehicles requires sharing of data between agents. This may require large amounts of bandwidth or clever data sharing algorithms (i.e., sharing large SLAM maps parts at a time). Besides determining which level of data and information abstraction is necessary to share in order to make the search more efficient (SLAM data, exploration graphs, positions, items in view, which room each agent is in, etc), a problem exists in utilizing these data on another platform with its own coordinate system. For example, a SLAM map in the frame of vehicle$_i$ will not be useful to vehicle$_j$ because the relative orientation and translation of each map is unknown, even if the relative scales, resolutions, probability ranges, and other specifics of the two maps are known.

The groundwork has been laid in GUST to allow multi-agent search without sharing the SLAM map and is described below. This is more practical for vehicles that do not have large capacity for processing, such as UASs. Suppose several vehicles are deposited and initialized in their arbitrary starting locations and orientations in front of the entry point to an area to be searched. The two vehicles plan on sharing their exploration graphs during the exploration process. For this, the relative translation and orientations of the two vehicles are required. If it can be assumed that the vehicles are both traveling through the same entry point, and their sensors are capable of determining the relative angle and center of mass of the entry point, these data may be used to transform between coordinate systems of both agents. For example, two quadrotors equipped with a LIDAR system entering a window may determine the transformation between each other's coordinate frames and thereby share SLAM map and/or exploration graph data, as well as be able to utilize the other's data. An algorithm is created to calculate the center of mass of the window in each frame by sampling the LIDAR data at the time of the calibration. Using the position of the vehicle in its frame at the time of calibration, the offset to the window is found. The same is done by the second vehicle. Assuming that the center of mass of the window is a unique feature of the environment, knowing these data provides an offset between the two vehicle's coordinate frames. The algorithm then calculates the slope of the window in each frame by randomly sampling points on the return around the window and determining an average slope to all other points in the LIDAR scan. A standard transformation matrix is thus generated using these data and may be used to directly transform SLAM map and graph coordinates. An example is shown in simulation in figure 7 .

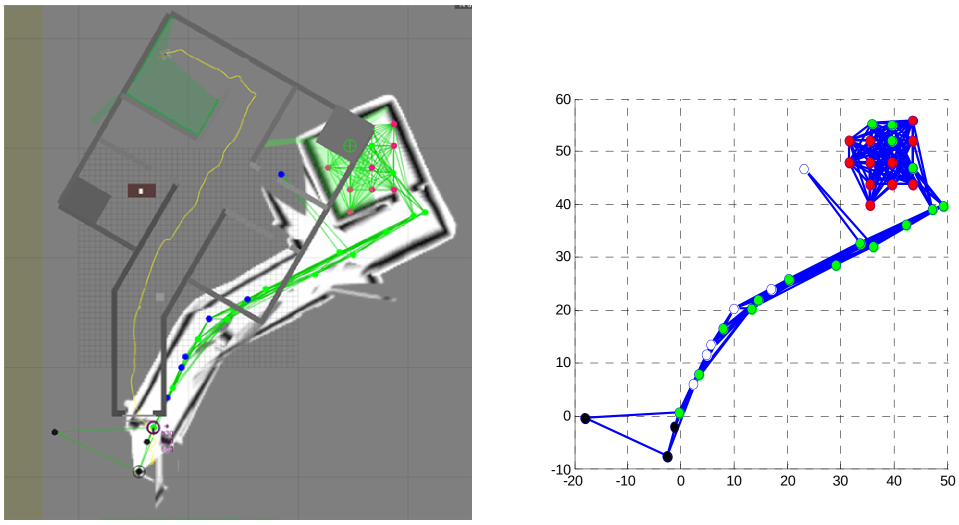

Figure 7. SLAM map (left) in vehicle coordinate frame overlaid by obstacles in physical frame, and corresponding exploration graph (right)

Here, vehicle$_2$ is rotated such that its x-body direction is not aligned with the map's north direction. Had the two directions been aligned, the map would be aligned with the geometry that it represents. The SLAM map in the vehicle's body frame is thus rotated when superimposed on the obstacles. As described, two such vehicles will produce similar maps in their own coordinate systems. The corresponding graph is shown in figure 7 on the right. Using the transformation described above, this exploration graph is transformed into vehicle$_1$'s frame and connected using strategies described in earlier sections. The process is shown in figure 8. The left-most graph is from vehicle$_1$, the center graph is the transformed graph from vehicle$_2$, and the right-most graph is the resulting cooperative, processed graph.

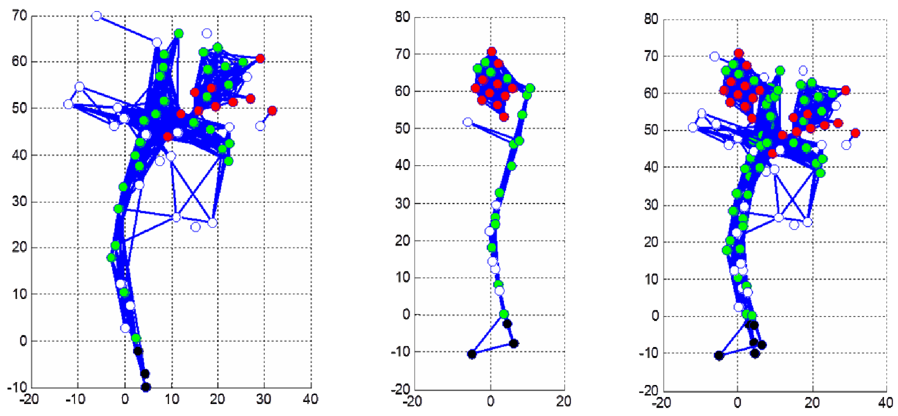

Figure 8. Vehicle$_1$'s exploration graph (left), vehicle$_2$'s exploration graph rotated into vehicle$_1$'s frame (center), and corresponding blended graph (right) in vehicle$_1$'s frame

This allows vehicle_1 to locate vehicle_2's position, for example, if vehicle_2 has located something of interest. This allows the vehicles to share coordinates for future work. For example, if the vehicles have heterogeneous set of effectors and vehicle 1 is required in the north-west room to do something that vehicle_2 is not capable of, it now has a path to get there without having to search and explore its way to the area.

If data transmission bandwidth is limited, the abstracted guidance data in the form of the exploration graph may be shared instead. The minimal SLAM map used for the GTQ takes up 160,000 bytes of RAM, and around 313,000 bytes with additional information. Transmitting the abstracted version as described, that is, the exploration graph, is smaller, depending on how complex the graph has become. After exploring about 1,300 square feet total, the graph of vehicle 2 in figure 8 is only 2,768 bytes. Vehicle 1's graph is 5,548 bytes. The entire composite graph is only 5% of the size of the minimal SLAM map. It is easy to transmit portions of the graph as well if required. Although the same could be done with the slam map, the partial information may or may not be directly usable as rapidly depending on which parts have been received.

Also, should a more complex guidance system be developed for multiple vehicle coordination, these data may be used to efficiently search a large area for a goal. A simple system might use the directional bias for multiple vehicles, which is now possible as a direction vector may be transformed into both frames. A directional bias may also be used to alter the weight of nodes in multiple vehicles' graphs should something of interest to the goal be detected, such a sign in the 6th IARC mission.

VI. Results

Simulation. GUST is used to simulate all data below. It allows all flight code to be simulated and then run on the actual vehicle. A simulated run of the 2011 IARC arena is shown in figure 7. The simulated truth vehicle is in the north east room after having traveled the yellow path.

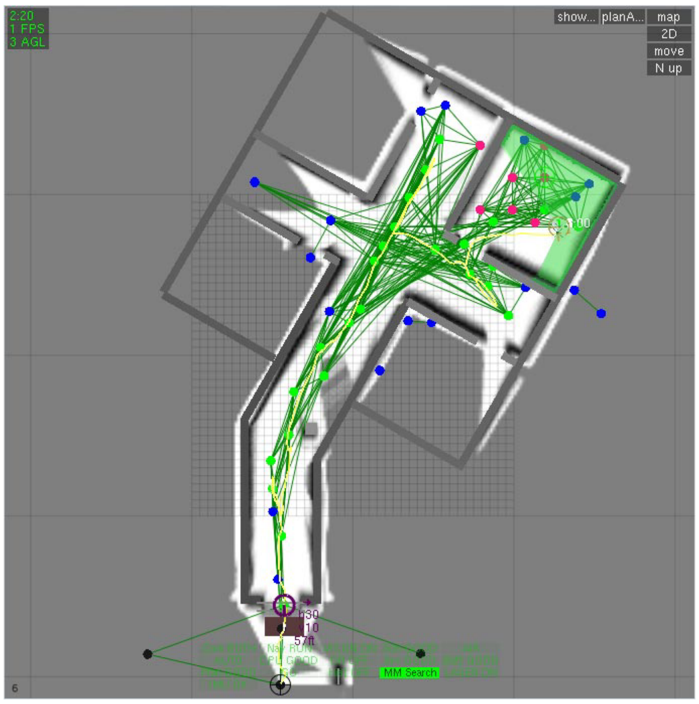

Figure 9. SLAM map and exploration graph overlay in GUST

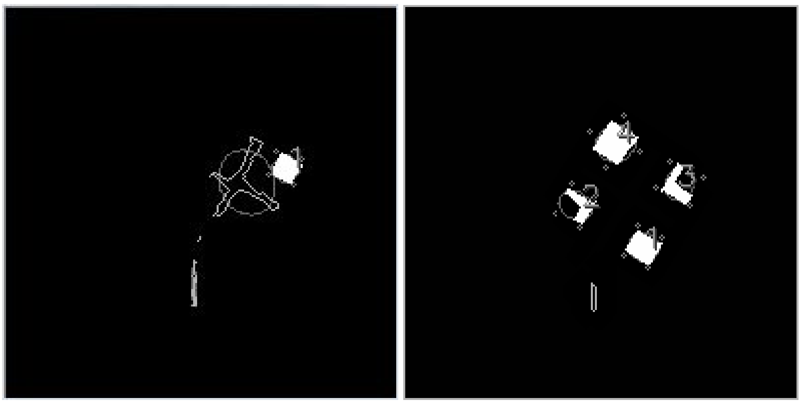

Figure 10. Room detection algorithm determines that a room exists in the north-east corner of the arena (left). Note that the vehicle in figure 7 has not yet scanned the remaining three rooms. Once scanned, all four are detected (right).

Although not necessarily the goal of this exploration system, entry into all four rooms is the metric used to judge particular SLAM-aided enhancements. Using a frontier only search algorithm in this arena does not guarantee a thorough exploration of the environment before vehicle resources are expended. This is generally due to exploring areas more than once as the SLAM map is not considered. Search with only graph pruning also does not guarantee entry into all rooms but dramatically enhances time spent in being stuck in local minima. By utilizing only graph pruning and path obstacle detection, 9 minutes and 50 seconds is the best time seen for entry into all four rooms without search. For the IARC however, this is around the limit of the flight battery and is not acceptable. With three enhancements described above, frontier generation, graph pruning, and path obstacle detection, all four rooms are entered with a nearly 100% success rate, generally in under 5 minutes. Figure 11 shows entry into all four rooms in under four minutes, the best time seen thus far. Room detection is not enabled to allow for timing comparison to the above metric. Once room detection is enabled, the result is seen in Figure 10 (right).

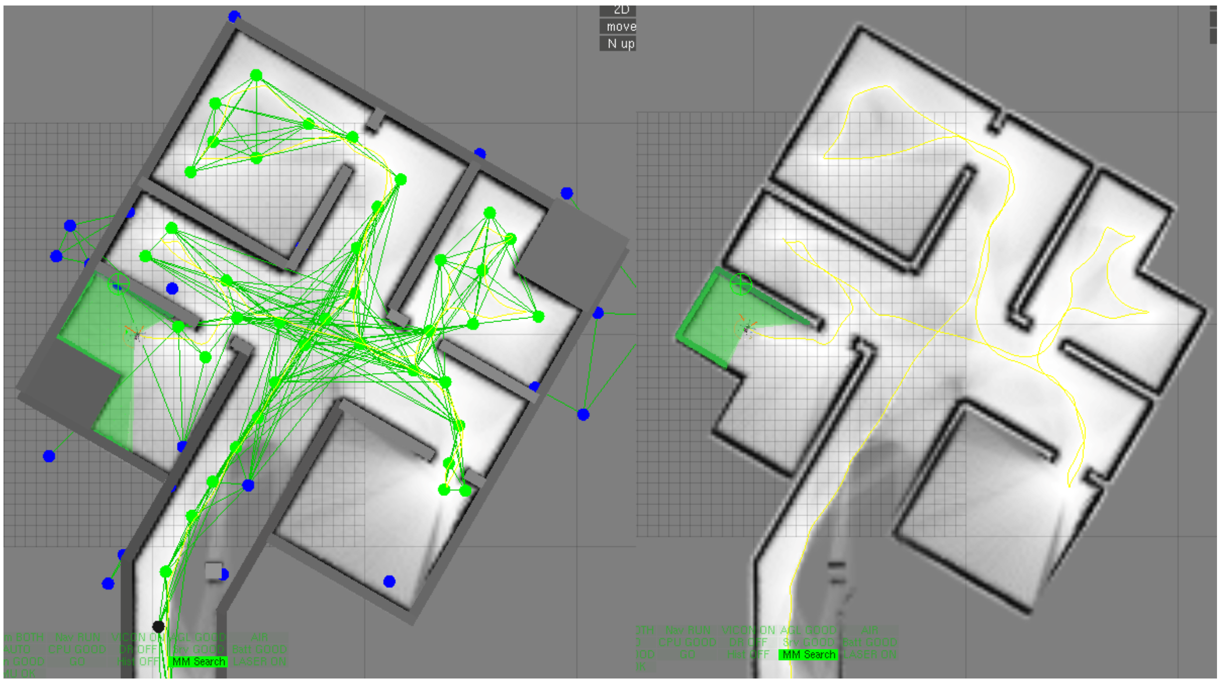

Figure 11. Four room entry in 3 minutes 42 seconds after takeoff. SLAM data, graph and simulated obstacles overlay (left) and SLAM with path history (right). Room detection is off.

Flight Test. This system was flown on the GTQ in the 2012 IARC in North Dakota. Since the time of the flight, Several guidance-specific difficulties were encountered when moving from the simulated to the competition environment. One concern was the performance of the SLAM upon which the both the navigation and guidance systems rely. In regard to guidance, rotational SLAM shifts may render entire sections of the environment unreachable as the SLAM map diverges from reality. Despite concerns, the SLAM system performed well during flight.

The guidance system was moved to a separate computer as a result of complexities with large graphs and potential memory allocation and access problems, large processing loads, and other potential sources of software failure. This was done so that in the event the software were to crash for any reason the navigation process would remain operational. Otherwise, had this not been done, the vehicle may have potentially lost control due to a guidance system and ultimately navigation system failure.

No guidance software stability problems were seen during the competition, although graph data transmission and recording systems did not perform adequately due to WiFi saturation problems in the arena. However, some data were recovered. Figure 12 shows the third (left) and fourth (right) of four attempts allotted to complete the mission.

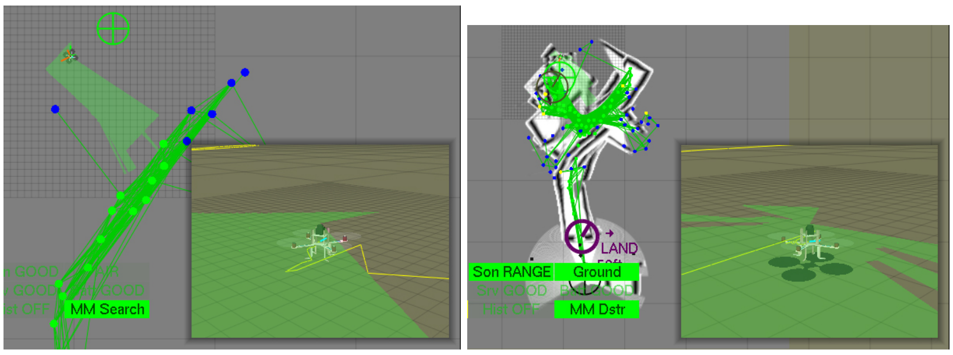

Figure 12. (left) Attempts 3 of 4 and (right) attempt 4 of 4 of the IARC

Attempt three of four did not provide SLAM maps to the ground station. In addition, the graph transmission to the ground station encountered issues partly as the vehicle made its way down the main hallway and into a room as can be seen in the figure. Data showed that the room was detected successfully and nodes were added inside as desired. The figure shows the vehicle inside the room containing the USB target, where the room is visible by the shape of the LIDAR scan. The flight ended in a crash about a meter from the USB stick due to a blind spot in the LIDAR and guidance which commanded the vehicle to back into a wall while capturing room nodes. At this time, obstacle avoidance only utilizes the current LIDAR scan. Code changes will be implemented to prevent the vehicle from moving backwards into the LIDAR's blind spot in the future.

Attempt four demonstrates the effects of rotational map shift on the exploration graph. It is clear that the vehicle would not have been able to use the SLAM map to exit the building as the north and south segments of the graph had no connection. The vehicle would have had to explore its way out using a directional bias toward the exit. The effect is seen in Figure 13.

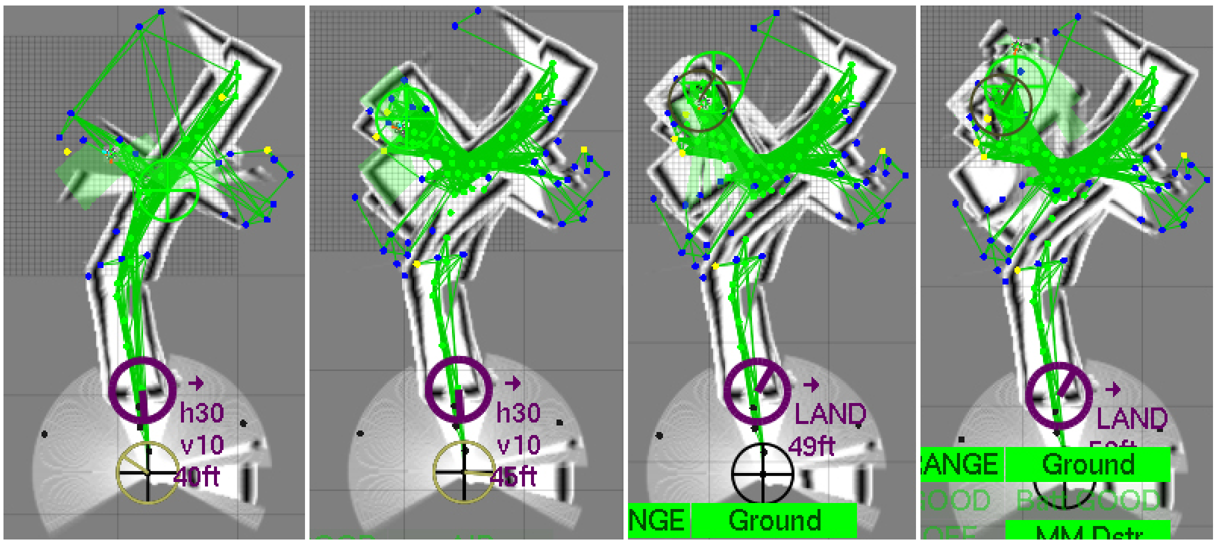

Figure 13 . SLAM map drift in the fourth IARC attempt. Note eventual disconnection of north and south graph clusters with rotational map shift

Updates to the CoreSLAM have had a positive effect on mitigating these effects. The resolution of the SLAM map has been artificially increased via interpolation which allows for better angular matching of scan to SLAM data.

VII. Conclusion

The guidance system performs well but with several limitations both in simulation and in actual flight. For example, the system described here relies on frontiers for exploration. Should a vehicle enter an area smaller than the radius of the LIDAR system, a sufficient number of nodes may not be added to thoroughly explore it. This effect was seen in the third attempt of the 2012 IARC. The security office offered no new frontiers as the vehicle entered it, as seen in figure 9 (left). Thus, without modifications, this node addition system may not be optimal for exploring small, confined spaces, but rather would work well in covering large areas.

Current work is underway to address the limitations of the guidance system described in preparation for the 2013 IARC. A tactical planner will be used to allow the vehicle to maneuver around obstacles with greater efficiency, and to allow it to reach areas where the line path does not perform adequately, such as around corners, for example. Voronoi skeletonization is also being studied as the SLAM map becomes solidified as an enhancement to the guidance system. Skeletonization provides guaranteed free paths inside the SLAM map should the exploration graph fail. Also, the SLAM map itself may be used as a frontier selection system to reduce the creation of erroneous nodes. A Kinect sensor and table finding algorithm will be developed to find tables for additional context, as it is known that the target resides on top of one. To ensure thorough exploration, the term "thorough" needs to be defined with respect to the goal of the search. If the goal is mapping of the interior, the SLAM map may be used to declare success. For the IARC, the camera's FOV will be considered when sweeping rooms to make sure all surfaces will be seen when searching for the target.

References

[1]M. Julia, O. Reinoso, A. Gil, M. Ballesta, and L. Paya, “A hybrid solution to the multi-robot integrated exploration problem,” Engineering Applications of Artificial Intelligence, vol. 23, no. 4, pp. 473 – 486, 2010.

[2]B. Yamauchi, “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.

[3]S. Moorehead, R. Simmons, and W. Whittaker, “Autonomous exploration using multiple sources of information,” in Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on, 2001.

[4]B. Kuipers, R. Browning, B. Gribble, M. Hewett, and E. Remolina, “The spatial semantic hierarchy,” Artificial Intelligence, vol. 119, pp.191–233, 2000.

[5]B. Kuipers and P. Beeson, “Bootstrap learning for place recognition.” AAAI/MIT Press, 2002, pp. 174–180.

[6]K. Konolige, J. Bowman, J. Chen, P. Mihelich, M. Calonder,V. Lepetit, and P. Fua, “View-based maps,” The International Journal of Robotics Research, vol. 29, no. 8, pp. 941–957, 2010.

[7]C. Stachniss, O. Mozos, and W. Burgard, “Speeding-up multi-robot exploration by considering semantic place information,” in Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on, May 2006, pp. 1692 –1697.

[8]F. Dellaert and D. Bruemmer, “Semantic slam for collaborative cognitive workspaces,” in AAAI Fall Symposium Series 2004: Workshop on The Interaction of Cognitive Science and Robotics: From Interfaces to Intelligence, 2004.

[9]2008 IEEE International Conference on Robotics and Automation, ICRA 2008, May 19-23, 2008, Pasadena, California, USA. IEEE, 2008.

[10]S. Ekvall, D. Kragic, and P. Jensfelt, “Object detection and mapping for service robot tasks,” Robotica, vol. 25, pp. 175–187, March 2007.

[11]C. Nieto-Granda, J. Rogers, A. Trevor, and H. Christensen, “Semantic map partitioning in indoor environments using regional analysis,” in Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on, 2010, pp. 1451 –1456.

[12]J. Maye, L. Spinello, R. Triebel, and R. Siegwart, “Inferring the semantics of direction signs in public places,” in Robotics and Automation (ICRA), 2010 IEEE International Conference on, May 2010, pp. 1887–1892.

[13]J. Larsson, M. B. Aless, and R. Saffiotti, “Laser based corridor detection for reactive navigation,” Industrial Robot, vol. 35, pp. 69–70, 2008.

[14]J. F. Diaz, A. Stoytchev, E. Stoytchev, and R. C. Arkin, “Exploring unknown structured environments,” in 14 th International Fairs Conference, Key West, FL, 2001.

[14] D. M. Sobers, G. Chowdhary, and E. N. Johnson, “Indoor navigation for unmanned aerial vehicles,” 2009.