Journal of Intelligent & Robotic Systems (2022) 106:58
https://doi.org/10.1007/s10846-022-01751-9

REGULAR PAPER

Online Aerial 2.5D Terrain Mapping and Active Aerial Vehicle Exploration for Ground Robot Navigation

Anthony Wagner[1] · John Peterson[1] · James Donnelly[1] · Shivam Chourey[1] · Kevin Kochersberger[1]

Received: 2 August 2021 / Accepted: 3 October 2022 / Published online: 25 October 2022

Abstract
This paper discusses a collaborative air-ground team of autonomous vehicles for exploring and navigating outdoors within an unknown environment. A custom multi-rotor equipped with onboard downward facing stereo cameras flies over an unknown environment capturing imagery and reconstructing a height map of the ground surface online. Together with a simple terrain classifier, this height map is used to detect obstacles and estimate navigation costs for a Clearpath Robotics Jackal. Given a user defined goal, an online exploration algorithm automatically directs the aerial vehicle to expand the frontiers of the map until a path for the ground vehicle from the current position to the goal is discovered. Simultaneously, this algorithm directs the ground vehicle to execute partial paths to make progress towards the goal before a complete path has been discovered. Aside from the trajectory execution and the state estimation for the unmanned ground vehicle and the graphical interface for the operator, the online exploration algorithm and reconstruction all run onboard the UAV by taking advantage of the graphics processing unit (GPU) computing capabilities of the Nvidia TX2 for both stereo calculations and map construction. This paper presents the improvements made to an existing multi-robot system, the mapping and exploration algorithms. Finally, this paper discusses both simulation and hardware experiments conducted to validate the behavior of the exploration algorithm.

Keywords Air–ground cooperation · Multi-robot coordination · Unmanned aerial vehicle · Unmanned ground vehicle

1 Introduction

A traditional ground robot has a variety of sensors for obstacle detection and avoidance. A few common types of sensors are cameras, lidar, and ultrasonic range finders. However, large obstacles or tall grass in the environment can obstruct the line of sight of these vehicle-mounted sensors. In many scenarios it would be useful if the unmanned ground vehicle (UGV) could have information beyond the field of view (FOV) of the sensors. An aerial vehicle with its own sensors can provide this additional information. UAVs can gather information faster than a UGV for three primary reasons. The first is that unmanned aerial vehicles (UAVs) typically have a higher top speed than a UGV of similar size. The second is the UAV is in the air and can fly above obstacles and take a shorter route. The third is the UAV can see larger areas from higher altitudes. These advantages are especially useful in a post disaster environment such as the one shown in Fig. 1. The large amount of scattered debris means a priori information would possibly not be accurate and thus would not provide a good route for a ground vehicle. In the same time required for the UGV to slowly traverse the debris, a UAV could explore a larger area and generate a better path for the ground vehicle.

This paper presents the significant hardware and software improvements made to an existing cooperative outdoor UAV/UGV system [20]. The previous system used a

This work was funded through the National Science Foundation I/UCRC Center for Unmanned Aircraft Systems Phase II Site Addition, Award No. 1650465. The Jackal ground robot was graciously loaned by the Army Research Laboratory for these experiments.

✉ Anthony Wagner
wagnera@vt.edu

John Peterson
jrpeter@vt.edu

James Donnelly
jd7@vt.edu

Shivam Chourey
shivam@vt.edu

Kevin Kochersberger
kbk@vt.edu

[1] Virginia Tech, Blacksburg, VA, USA

Fig. 1 Severe damage caused by a tornado. A priori information about the environment would be out of date due to the large amount of debris [7]

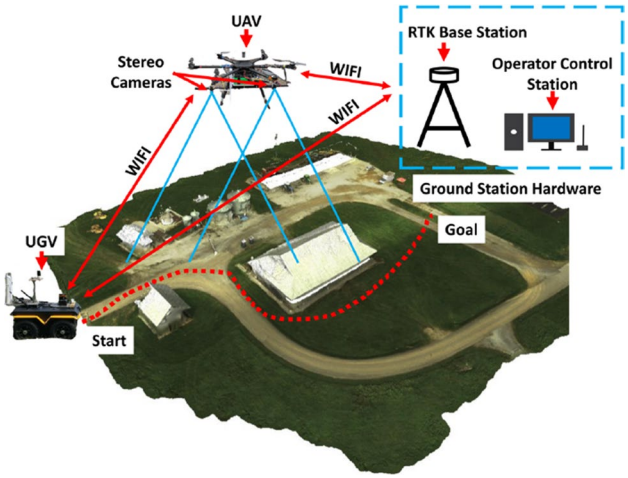

Fig. 2 Overview of the system shows a UAV and UGV collaborating to route a UGV in an outdoor environment

monocular camera and color segmentation to detect obstacles. The goal of the collaborative air-ground team is to explore and navigate an unknown outdoor environment as shown in Fig. 2. The objectives of the improvements outlined in this paper were to provide better obstacle detection and to add the exploration capability the previous system lacked.

There are several improvements presented in Section 2, with the major improvements being the addition of a stereo camera, a new mapping algorithm (Section 4), and a novel exploration algorithm (Sections 5 & 6). The addition of a stereo system adds the ability to detect obstacles based on terrain height. The mapping algorithm uses the GPU on the onboard computer to more quickly process the data from the cameras. The exploration algorithms focus on how to explore an unknown environment with a UAV to find a route for a UGV. This includes the development and comparison of two different UAV exploration methods and a method of giving the UGV partial plans during exploration to maintain operation tempo.

1.1 Related Work

The related work outlined here is organized based on these five main relevant categories: mapping with UAVs, stereo-cameras, exploration methods, frontier exploration, and similar systems respectively. Aerial data has been used in several systems for mapping as well as some others that also used it for path planning for a UGV. Kim [15] presented a UAV/UGV system that uses a stereo vision system on the UAV to detect obstacles for the UGV. This system also used markers on the UGV to allow the UAV to localize in a GPS denied environment. Miki presented a novel cooperative UAV/UGV system that included mapping the environment for UGV path planning [18]. This system used a monocular camera and time of flight sensor to map an indoor environment. Kim [16] presented a system where a UAV initially maps an area so that a UGV equipped with a laser scanner can construct a final, higher fidelity map. Schneider presented an online mapping system for a UAV using RTK GPS and SLAM [23].

Warren noted the importance of alignment of long baseline stereo cameras and how vibration and knocks can force a re-calibration of a stereo-camera and presented an online method to re-calibrate the camera [26]. Rovira-Más looked at combinations of lens focal lengths and baselines for long-baseline stereo cameras based upon the necessary range of use [21]. Milella used a multi-baseline stereo camera to perform 3D scene reconstruction and segmentation with both long and short range obstacles [19]. Haubeck used an off the shelf stereo camera on a UAV for mapping archaeological sites [10].

Juliá et al defined autonomous exploration as "The ability of mobile robots to autonomously travel around an unknown environment gathering the necessary information to produce a useful map for navigation." According to Juliá, the term exploration contains path planning but also can be used to describe techniques that would not fall under the umbrella of path planning [13]. Exploration methods can focus on minimizing exploration time or focus on developing better maps. Cesare [4], Ferranti [8], and Kalra [14] did work on leveraging the additional vehicles in multi-vehicle exploration systems.

The frontier based exploration method was introduced by Brian Yamauchi [27]. Frontiers are defined as the "regions on the boundary between open space and unexplored space." Frontiers help address the issue of where to go in an unknown environment to gain as much

information as possible. To detect frontiers Yamauchi proposed using "a process similar to edge detection and region extraction in computer vision" to find these boundaries. To select the best frontier to visit, Yamauchi proposed sending the robot to the nearest accessible and unvisited frontier. Bachrach [1] proposed a method of weighted frontiers which aimed to maximize the amount of space explored while still localizing accurately with simultaneous localization and mapping (SLAM). Accurate localization requires keeping a certain number of recognized landmarks within sight. Therefore, sending the robot too far into unexplored space would harm localization and mapping accuracy. Bachrach used a weighted method to better select a frontier to visit from the identified frontiers [1]. Burgard [3] used a weighted cost of reaching a frontier with its expected benefit. Gonzáles-Baños and Latombe used the same two criteria of distance and benefit but combined them using an exponential function [9]. Umari presents a method using RRTs to explore an unknown environment [25]. The frontier detection is done by checking where the expanding tree crosses into unknown space. Additionally, both a global and local tree are used to find local and global frontiers. This is a similar method to what we propose but differs in our exploration is targeted towards a global goal and fast exploration time while Umari's method is targeted at exploring the entire environment.

Shim [24] combined local obstacle map generation with an exploration algorithm based on model predictive control (MPC) to explore simulated urban environments. Their exploration was to find a path for a large UAV (Yamaha RMAX) through a dense unknown urban environment. They used a cost function that minimized penalties for tracking and obstacle avoidance. There has also been some work with collaborative UAV/UGV exploration. Hood [11] presented a UAV that would follow a UGV in an indoor environment and aid in situational awareness with its better vantage point. An augmented reality (AR) tag was used for the UAV tracking the UGV and SLAM was used to localize and map the environment for the exploration. Salas presents a UAV UGV collaboration tested in simulation where the UAV follows a predefined flight plan to map the environment and search for the goal target. Once the target is found by the UAV the UGV is sent sub-goals generated from a global planner on the map generated by the UAV [22].

Our work brings together some of the ideas presented in these other works into a large system that is used in a real-world outdoor scenario. It also provides a novel targeted frontier exploration method similar to Umari but with improvements towards decreasing exploration time and specializing it towards exploring a route for a UGV.

2 Hardware Improvements

The hardware for this system consists of four major subsystems: the UAV, the UGV, an RTK GPS system, and the Operator Control Station (OCS). The UAV is a Tarot 960 hexacopter equipped with a Nvidia TX2, a Pixhawk Cube flight controller running Arducopter, and a custom long-baseline stereo camera. Figure 3 shows the UAV used in the experiments and Fig. 5 is a close up of one of the cameras in the stereo pair. The UGV is a mostly stock Clearpath Jackal (Fig. 4). It is important to note that there is a Hokoyu lidar and a webcam on the vehicle but they are not used for obstacle detection/mapping. For a base station a simple desktop is used for visualization and mission control since the computation happens on-board the UAV (Fig. 5).

Fig. 3 Tarot T960 hexacopter with custom stereo system and Nvidia TX2

Fig. 4 Clearpath Jackal used for these experiments

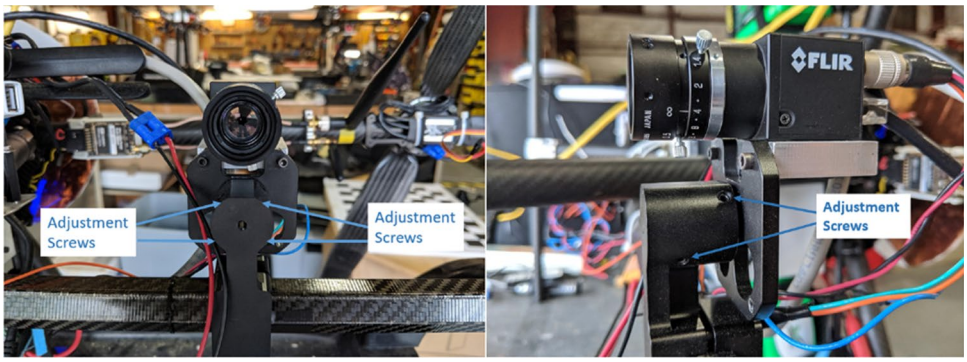

Fig. 5 One of the FLIR/Point-grey cameras mounted on the boom. Mount allows for adjustment which is critical for proper calibration and use

2.1 Stereo Imaging

One of the major improvements to the system was switching from a monocular camera to a custom long-baseline stereo camera. The stereo camera allows for the terrain depth to be measured without using a more expensive sensor such as lidar. The first iteration of the stereo camera used USB3 cameras which were found to produce a lot of broadband noise that increased the noise floor significantly on our GPS signal and resulted in poor GPS performance. Davuluri and Chen explored the root cause of this phenomena [5]. In order to get rid of that interference FLIR Blackfly S GigE cameras were selected for their Ethernet interface and existing ROS driver. These were mounted to a carbon fiber tube using a custom clamp and adjustment mechanism. The cameras were separated to form a 600 mm baseline which was selected based on a target mission altitude of 12-25 m.

The key processes for using a long baseline stereo camera are alignment and calibration. We found that getting the optical center of each camera's image within $1-2$ pixels and getting minimal roll between the cameras was key to getting a good calibration and disparity image. Alignment was achieved by pointing the cameras at a distant object ($\sim>$ 100 m) that would have a disparity of less than one pixel, as shown in Fig. 6. The disparity is the horizontal pixel distance between a corresponding pixel in the left and right image. A disparity of less than one pixel indicates that an object is sufficiently far enough away that it shows up in the same pixel location in both the left and right image. The camera adjustment screws were then adjusted until the red center markers overlaid on both images aligned with the same point on the distant object. To calibrate the stereo camera the ROS *camera_calibration* package was used. This package provided a convenient GUI to capture the calibration data and then generate the calibration.

One of the primary reasons for selecting the Nvidia TX2 as the computer on-board the UAV was for its GPU. We selected a GPU-based implementation of Semi-Global matching algorithm made by researchers at Universitat Autonoma de Barcelona [12]. On our system with 1024×768 images we saw frame rates of $\sim 5-7$ frames per second. Figure 7 shows an example disparity image generated by the system.

2.2 RTK GPS

Both UAV and UGV used SwiftNav Piksi Multi RTK GPS sensors for precise localization. For the UAV, the RTK GPS feeds into the Pixhawk flight controller where it is combined with measurements from an inertial measurement unit, gyro, and magnetometer. The resulting state estimate from the flight controller is then used by the ROS system. For the UGV, an extended kalman filter (EKF) combines the RTK GPS measurements with measurements from the built-in inertial measurement unit (IMU) and wheel encoders. Both vehicles operate in the same map frame and use their state estimates to localize themselves in this frame. The single band Piksi RTK GPS had been used in earlier versions of this system. It had reliability issues because of the use of serial radios to pass RTK corrections, and longer times to converge to an RTK fix. The new system uses the same dual-band Ubiquiti WIFI radios used for primary communication

Fig. 6 Example of how the UAV is positioned for calibration and example images showing the alignment process

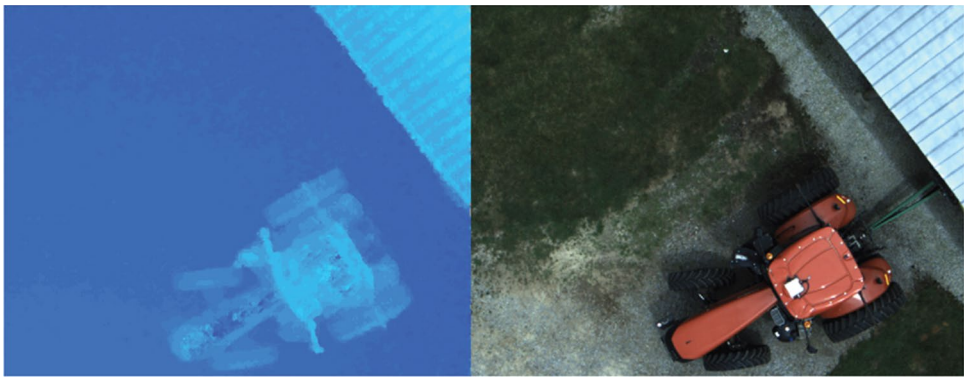

Fig. 7 Disparity image (left) captured using the stereo system at an altitude of ~25 m. A large tractor and part of a building can be seen in the raw color image (right)

between the vehicles for the corrections. While this has a reduced range, we observed better reliability and reduced latency in the corrections.

3 Online Stereo Extrinsic Calibration

One key feature added to the stereo mapping system on the UAV is the ability to calibrate the extrinsics of the stereo camera online. The extrinsics calibration for the stereo camera is the measurement of the position and orientation of the stereo camera relative to the base frame of the UAV. Previously this calibration had been manually measured/estimated. However, small inaccuracies in the estimated orientation resulted in large projection errors of obstacles into the costmap. A costmap is a map of the environment divided up into grid cells where each cell represents the cost (difficulty) to traverse that cell. In multiple previous tests, obstacles were shifted in the costmap by up to a few meters.

Coloring the top of the RTK base station antenna with a recognizable color makes the antenna detectable in stereo images. Orange was selected to contrast against a background of grass. Using the antenna is advantageous since we know that this location is (0, 0, 0) in the map frame. By moving the UAV through a suitable trajectory with the antenna in view, we can build up a set of point correspondences to estimate the relative position and orientation of the camera to the UAV's frame.

The antenna was detected using simple process with OpenCV. First the image was thresholded in the HSV (Hue Saturation Value) color space. Contours were extracted and then the minimum enclosing circle was found for the largest contour. The center of the minimum enclosing circle gives us the antenna position in the image. The coordinates of the antenna in the camera's frame of reference can then be estimated using the stereo parameters.

For each estimate of the antenna's location in the camera frame from the stereo camera, the projected location of the antenna in the camera's frame based on the UAV's state estimate is recorded. Each measurement creates a point correspondence between the antenna location estimated with the camera and the location based on the flight controller state estimate. These are stored as two separate point clouds. With these point correspondences we use the Point Cloud Library's TransformEstimateionSVD to estimate the rigid body transform between the two point clouds which gives us an estimate of the transform between the flight controller and the stereo pair.

Calibration imagery was captured by manually ascending and descending the UAV above the antenna while yawing. This flight pattern was found to produce satisfactory calibrations.

To test the extrinsic calibration points easily identified in the costmap were surveyed using an RTK GPS. These points were a portable hand washing station (1), the front right tire of a tractor (2), the north corner of a barn (3), and the corner of a concrete pad (4) [see Fig. 8]. The position of each point was measured for one minute. The average position of each point was projected onto the costmap for visual comparison. Due to the discontinuities in the map it would be difficult to quantify the error exactly but the image shows that the calibration has a projection error of less than 0.5 meter. This is a huge improvement over what we observed before adding the extrinsics calibration which sometimes

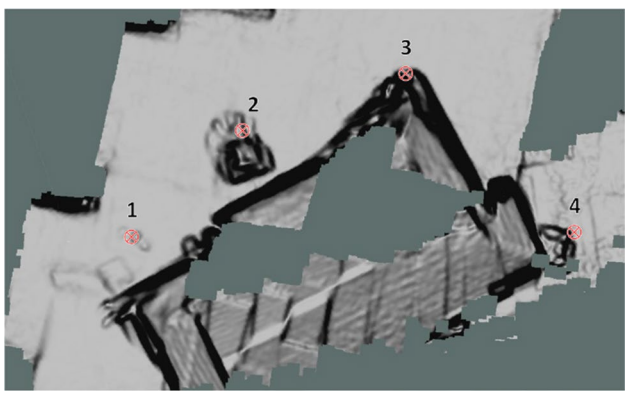

Fig. 8 Costmap with RTK surveyed points marked. Dark areas represent areas of high cost (obstacles)

had a projection error of multiple meters. A 0.5 meter accuracy places a lower bound on the size of obstacles the system can successfully avoid of to around a meter.

4 Online Mapping

The mapping subsystem takes in disparity information and the raw color images to produce costmaps for the path planners. Figure 9 shows the overall flow of data through the mapping subsystem. Color imagery and disparity generated by the stereo camera combine to generate the different maps shown. Map dimensions and spatial resolution (m/pixel) are fixed at launch so operators must predefine the operating area and resolution. A height map was selected for efficiency with a top down aerial view of an outdoor environment. This representation is significantly more compact than a dense 3D voxel representation, however it is unable to model overhangs. A GPU based implementation converts the computed disparities and color images into a colorized point cloud in map coordinates using the state estimate provided by the flight controller. The map frame the point cloud is projected into is a global frame based on the RTK GPS localization. The flight controller's estimate of position is accurate due to the use of RTK GPS. However, we have observed inaccuracies in the constructed map due to poor estimates of roll and pitch. Donnelly has done work on using feature matching to improve the registration of the mapping in his Master's thesis [6]. His thesis work was done with the same system but was not implemented at the time of this work.

First the points from the colorized point cloud in a one meter radius of the position of the UGV at the time the image was captured are filtered out to ensure the UGV does not show up in the map. The filtered points from the colorized point cloud are then accumulated in three separate dense grid maps. The first accumulates the weighted sum of z height of points falling into each grid cell. The second accumulates the weighted sum of RGB color values of points falling into each grid cell. The third accumulates the total point weight within each cell. Each point is assigned weight proportional to its inverse depth to reflect the loss of accuracy in z height and loss of detail in color for more distant points. From these grid maps we may distinguish between observed and unobserved space as well as compute the average height and average color of each observed cell.

These GPU data structures were converted into a costmap at the request of the exploration software. The costmap consists of two terms, an obstacle layer based on estimated surface angle and the second layer based on a terrain cost. To generate the obstacle layer, we begin by computing the weighted average height in each cell. The average height image is blurred by a Gaussian kernel with a user set standard deviation. This blurring step removes spurious discontinuities in the height map at the edges of projected images due to errors in the estimate of the orientation of the UAV. This affects the size of obstacles the system is sensitive to since all obstacles are dilated by this step. After the blurring step, the resulting height image is eroded to account for the unobserved areas of the map.

The Scharr gradients of the height are calculated in the x and y directions. This magnitude of gradient is used to compute a measure of the surface angle using (1). The result of this step is shown in Fig. 9 as the gradient magnitude.

$$\theta(i,j) = \text{atan2}(\|\nabla(i,j)\|, 2 * \text{resolution}), \tag{1}$$

Where $\|\nabla(i,j)\|$ is the magnitude of the gradient in cell i, j, $\theta(i,j)$ is the estimated surface angle and where *resolution* is the resolution of the map in m/pixel. The factor of 2

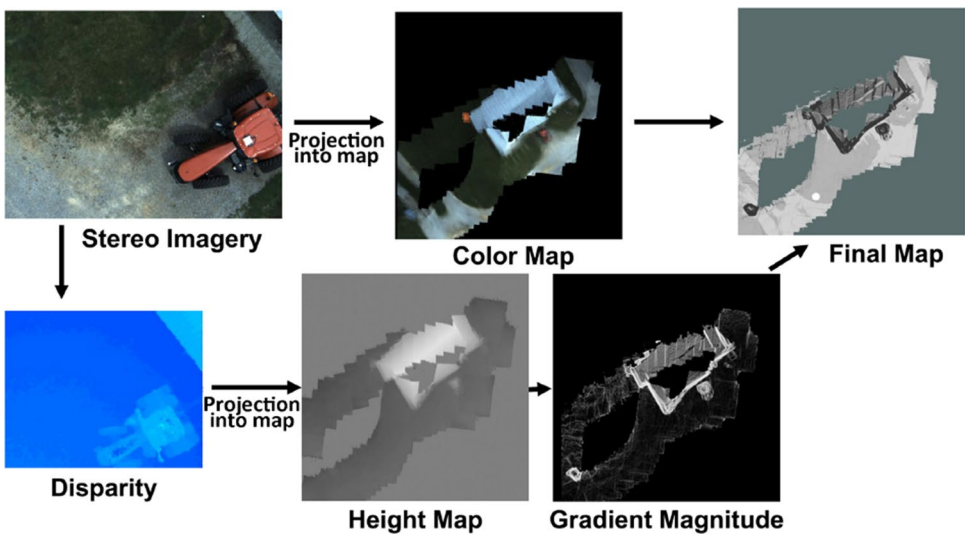

Fig. 9 Data flow from stereo imagery to final costmap

represents the distance between the centers of the cells in the 3×3 kernel. Then this surface angle is converted into a cost. We simply use a clamped quadratic function relating angle to cost using (2).

$$C(i,j) = C_{lethal} * \min\left(\left(\frac{\theta(i,j)}{\theta_{lethal}}\right)^2, 1.0\right), \quad (2)$$

Where $C(i, j)$ denotes the computed cost, C_{lethal} is the cost value denoting a lethal obstacle (100.0), and where θ_{lethal} is the user set maximum angle set to 60 degrees.

The terrain layer uses a simple 2-class, HSV terrain classifier to compute a terrain cost. After computing the average RGB color for each cell, then converting to the HSV color space, cells with a saturation or value below user set thresholds are classified as road, otherwise the cell is classified as off-road. If a cell already has a valid slope cost, a terrain class specific cost is added to compute the total cost of the cell. Off-road is assigned a cost of 27 while road is assigned a cost of 16 out of the total 100. These costs were arbitrarily assigned based on human perceived cost and the effect of different values was not evaluated in this work. We avoid applying the terrain cost for cells where the slope cost was not successfully computed to avoid underestimating the cost near the edges of explored space. The total costmap provides both information about the cost of traversal and describes what areas of the environment have already been explored. This map is then passed on to the navigation stack for exploration and planning.

5 Online Exploration with UAV

One of the primary goals of this work was to develop an online exploration algorithm for directing the UAV to explore the environment for the UGV. This exploration algorithm needed to run onboard the UAV, and it needed to find a path for the UGV in a reasonable amount of time. Two exploration algorithms were developed and tested. The first method, called the Dijkstra Frontier method, plans a path between the start and goal, and moves the aerial vehicle to the frontier between observed and unobserved space. The second method, called RRT Exploration uses a bi-directional Rapidly-exploring random tree (RRT) to identify multiple candidate goals for the UAV to select between. Both methods use a targeted approach to exploring the environment as compared to an exhaustive search. More exhaustive methods such as using a raster pattern would take longer when UAV time is at a premium. Additionally, a simple pattern does not leverage the information gained during exploration.

5.1 Dijkstra Frontier Exploration

The first method explored was the Dijkstra Frontier method. This method would be similar to using D* or re-planning with Dijkstra/A* onboard the UGV without any UAV [17]. Traditional exploration with a UGV and D* or re-planning would involve updating or re-planning every-time the costs in the map update and would plan from the current location of the UGV. The difference in this system is that the UGV is not doing the exploring but rather the UAV. For this system the UAV is operating at a predetermined altitude that operators have determined is free of obstacles. This means that the UAV can explore anywhere in the environment without concern of running into obstacles. Exploring with the aerial vehicle presents several differences from exploring with a UGV. First the UAV is not bound by the same constraints of obstacles as a UGV. One example of this is a UAV can sense an obstacle in front of it on the ground but it can go over the obstacle without collision. Exploration can benefit from going over obstacles and not trying to avoid them but this can also lead to the UAV traveling past an obstacle and exploring a region disconnected from the explored area containing the UGV. This would prevent the system from finding a valid path for the UGV. The aerial perspective of a UAV allows for some different approaches to be taken.

With the previous considerations taken into account the Dijkstra Frontier method was developed. Plans are generated from the current UGV location so that the frontiers found will always be impeding the valid path. Doing this makes it such that there is always an already explored path leading up to the new frontier. To find where the frontier is located, we generate a plan using an optimal planner (in this case A*) and iterate through the plan generated until the first point in unexplored space is reached. This point is then labeled as the frontier. To maximize the size of the new area explored, the UAV's goal is offset by a fixed distance along the vector between the UAV and the frontier. If the UAV went exactly to the frontier point then half of the field of view (FOV) of the stereo system would be in known space. This would result in longer overall exploration times with approximately twice as many exploration cycles as the UAV. After the UAV reaches its goal, the UGV path planner is run with unknown space represented as obstacle/impassable. If it returns a valid plan exploration is done. If not, the cycle is repeated. A visual of the algorithm in action is shown in Fig. 10.

5.2 RRT Exploration

The RRT Exploration method was developed due to some of the observed issues of the Dijkstra Frontier method. The main issue was that in certain environments the exploration would oscillate between exploring two sides of an obstacle and this made the UAV travel a longer than necessary

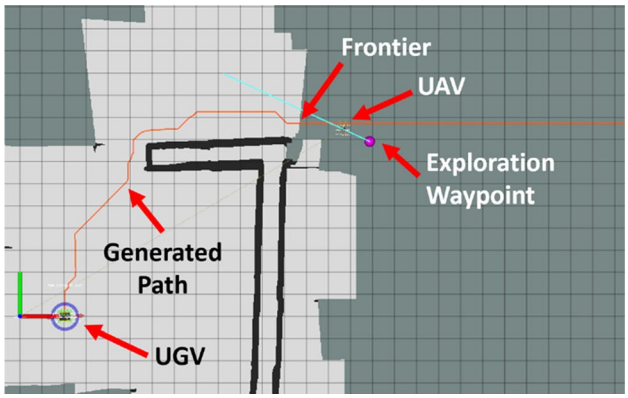

Fig. 10 Diagram of the Dijkstra frontier method visualized in RVIZ

distance. An example of this is shown in Fig. 18. The Dijkstra frontier method shows a lot of activity going between the two sides of the first obstacle. This method attempts to address the issue by using RRTs to generate multiple sub-optimal paths so that the UAV can select frontiers closer to the UAV.

The primary reason that a RRT was selected is that as a sampling based path planning technique sub-optimal paths can be generated. The problem with the Dijkstra frontier method was that Dijkstra and similar methods provide an optimal path and have no way of generating multiple sub-optimal paths in the environment. The resulting paths can change dramatically when a little bit more of the environment is explored. Generating multiple sub-optimal paths allows the UAV to select closer frontiers even if the UGV path is slightly longer.

To find the frontiers, a bi-directional RRT in the two-dimensional costmap is used to search from the goal and the UGV start point, where the RRTs are constrained to unknown and known space respectively. The possible connections of the two RRTs will correspond with the frontiers. Since the known space will only expand, and assuming a static environment, the RRT inside known space is expanded on each cycle rather than being regenerated from scratch. The RRT in unknown space is regenerated each cycle for simplicity. For the size of environment explored in this experiment there were no calculation time issues with just regenerating the unknown RRT. Regenerating the unknown RRT could be replaced by using a method to update the tree to remove branches that now intersect with found obstacles and known space.

The next section will detail the specifics of the RRT exploration method as it is implemented. It follows similar logic to the Dijkstra Frontier method as it goes to frontiers until a valid path for the UGV goal is found. Figure 11 shows how the RRT explore algorithm fits into the higher level logic. The function call to generate a plan is provided with the following information: the number of iterations for adding nodes in the unknown tree, the number of iterations for adding nodes to the known tree each cycle, the obstacle map, the UGV goal, and the current UAV position.

5.2.1 Mask Generation

The first step in the RRT explore algorithm is to generate some binary masks to help the algorithm later quickly identify certain regions. The three masks generated are known space, unknown space, and obstacles. These are implemented as OpenCV masks so that they can be used with other OpenCV functions.

5.2.2 RRT Generation

The RRT generation follows a generic RRT implementation. To start, a point is selected as the root of the tree to which all branches will eventually lead back to. A random point is sampled inside the region of the obstacle map. The nearest neighbor to this point is found using a simple search since the number of points is relatively low. The RRT implemented uses a fixed length branch. This means that each branch added is a specified length in the direction of the sampled point from the sampled point's nearest neighbor. This improves how well it can expand in tight spaces as longer branches are more likely to not be added due to collisions. Before the branch is added to the tree, it is collision-checked with obstacles. This is done using the masks previously generated. Using the OpenCV line iterator the pixels of the branch are iterated through and if any go outside the mask a collision is detected. If no collision is detected the branch is added to the tree. Each time a node is added its distance to the root of the tree is stored for quick distance checking later.

5.2.3 Unknown Area RRT Generation

The second step after the mask generation is to generate the RRT in the unknown area. Since the unknown area is shrinking the RRT cannot be regenerated without an algorithm to collision check the existing tree with areas that were updated. Thus, for simplicity's sake the RRT in unknown space is entirely regenerated. This has some impact on how fast the algorithm runs but the tree is able to expand quickly in the unknown space since there is more space for it to operate. The unknown tree is generated with the root of the tree at the goal defined for the UGV and expands outwards. The number of iterations used to add to the unknown tree ranged between $500 - 2000$ during the trials for this research. This number can be selected for other trials based on the size of the environment and performance considerations.

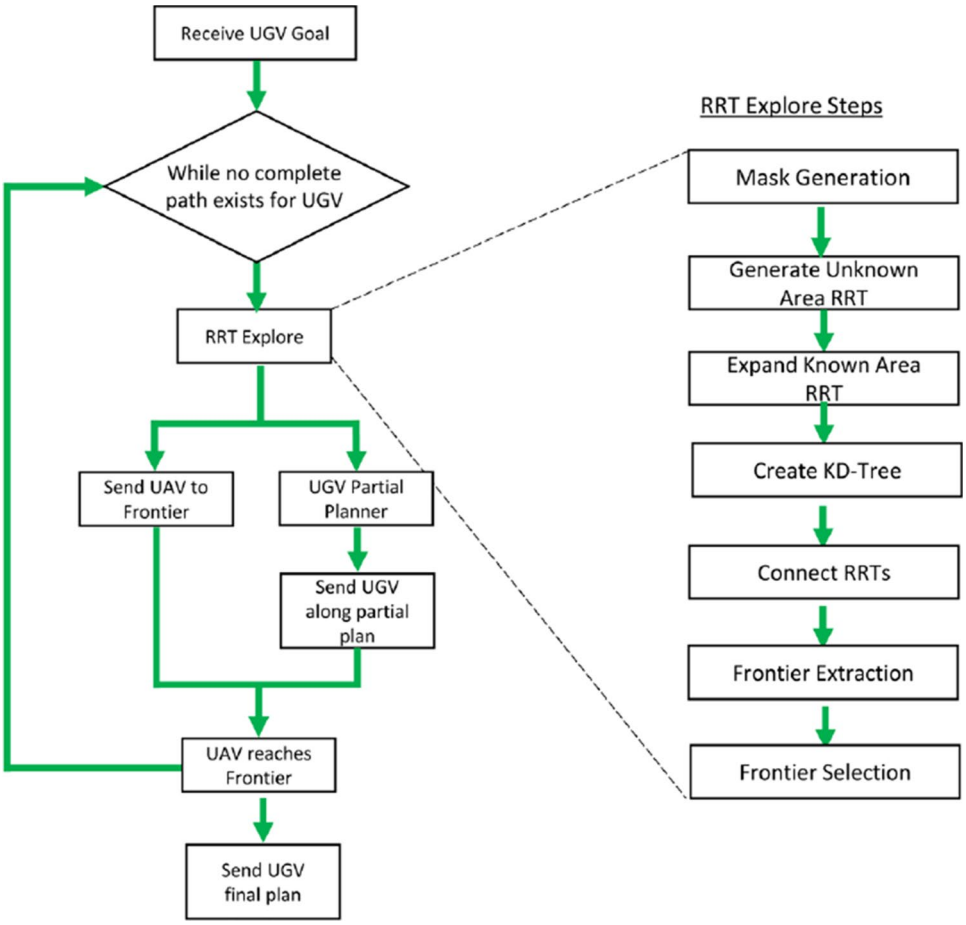

Fig. 11 Flowchart for how the RRT explore algorithm fits in the system and the specific steps of the RRT explore algorithm

5.2.4 Known Area RRT Expansion

The third step is to expand the known tree. This follows the same basic RRT algorithm as the unknown tree, but is inside the known space and is expanded instead of regenerated. The known tree is able to be expanded since the cleared space in the known area will not change since this is assumed to be a static environment. Thus, all existing branches in the tree will remain valid for the lifetime of the exploration. Since the known space is smaller and will have obstacles in the way, more iterations are required to find a valid path between two points. Expanding with each cycle spreads the computation over the life-cycle of the exploration. Figure 12 shows the environment after both trees have been generated.

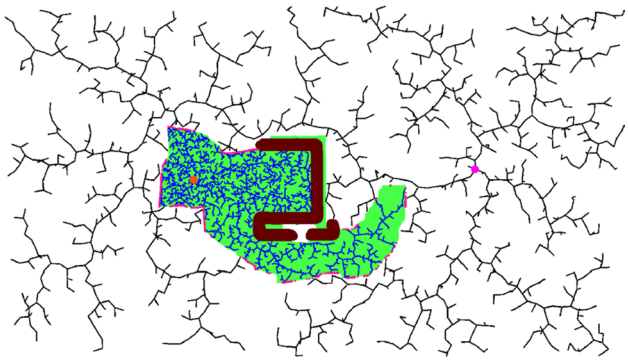

Fig. 12 RRT in unknown space (white area) expanding around the known space (green area) and obstacles (dark red). Start is the orange circle and the goal the is pink circle

5.2.5 KD-Tree Generation

The fourth step in the process is to generate a two-dimensional KD-tree for the points in the unknown RRT. A KD-tree is a data structure that stores points in k-dimensional space and allows for quick and efficient nearest neighbor searches [2]. This will be used in the connection phase of the algorithm as a large number of nearest neighbor queries are going to be made. The nodes in the two-dimensional KD-tree are the x and y coordinates of the nodes in the unknown tree.

5.2.6 RRT Connection

The fifth step is connecting the two RRTs. Connecting the two trees will find the possible paths in the environment. Due to the large number of nodes in both trees there are a large number of possible connections. To reduce this to a more reasonable amount, a single "best" connection is generated for each node in the known tree. For a node in the known tree its nearest neighbors in the unknown tree are identified using the KD-tree. Nearest neighbors are determined using a fixed neighborhood size. Once nearest neighbors are determined, each connection to the neighbor is collision checked and ones that result in a collision are removed. For the remaining connections, a cost function is used to determine the 'best" connection. The function is a weighted sum of the distance through the tree from start to goal and the distance between the two nodes. The distance between the start and goal is calculated using the root distance element in the two nodes that was saved during tree generation. The weighting used in this implementation was a weight of 0.1 for the total distance and 1.0 for the distance between the two nodes. This weighting scheme was determined to find the balance between finding the shortest overall path while also making sure it is a reasonably close connection. This step of connecting is repeated for each node in the known tree to find multiple paths.

5.2.7 Frontier Extraction

The sixth step is to extract the frontiers from the connections. The point where the connection crosses from known to unknown space is the frontier. To extract the frontier, we iterate along the connection from the known tree until an unknown pixel is reached. This transition point is that connection's frontier.

5.2.8 Frontier Selection

The seventh step is to select the best frontier for the UAV to visit. A cost function is used to select the best frontier. The cost function is the weighted sum of the straight-line distance to the goal and the straight-line distance to the UAV. These two parameters are used to balance between finding the shortest path for the UGV and minimizing how much the UAV has to travel. If the weight for the straight-line distance to the goal (w_g) is larger, the exploration will favor exploring frontiers closest to the goal and if the weight for the straight-line distance to the UAV (w_u) is larger the exploration will favor frontiers closer to the UAV. Using static weights would work but would not account for how the importance of the two variables change as the exploration frontier gets closer to the goal. When closer to the goal, the distance to the goal is more important than the distance to UAV. When the exploration is near the beginning, reducing UAV travel distance is more important. To handle this the weights are determined based on how close the frontier is to the goal. The selected weights, as a function of frontier distance to goal, are shown in Fig. 13. For the weight functions, a logarithm was chosen for the goal distance weight as its larger near the goal and fades out as we get away from the goal nonlinearly. A linear ramp that saturates was chosen for the UAV distance weight as we want it to increase as we get away from the goal but not increase exponentially once we reach a certain point. Once in the saturated area the cost will just increase linearly. There is no maximum distance from the goal that it will not consider. The exact weights and equation parameters were selected based on empirical observations and tuned manually based on watching the performance of the exploration in sample environments in simulation. The performance of the weighting algorithm in larger environments was not tested since the UAV endurance would not permit exploring larger areas. These values were observed to generally work well across the size environments tested in this work. If a UAV with longer endurance was to be used in a larger environment, the main parameter to look at for tuning is the distance from the goal at which the UAV distance weight saturates. The equations for the weights are below with the selected values used. w_g is the weight for the distance between UGV and goal and w_u is the weight for the distance between UAV and goal.

$$w_g = -0.1 * log(EuclDist(goal, ugv)) + 1, \tag{3}$$

$$w_u = min((0.027) * EuclDist(goal, uav) + 0.2, 1.0), \tag{4}$$

Using the weights from the formulas the frontier cost function is calculated as shown below.

$$cost_{frontier} = EuclDist(goal, ugv) * w_g + EuclDist(goal, uav) * w_u, \tag{5}$$

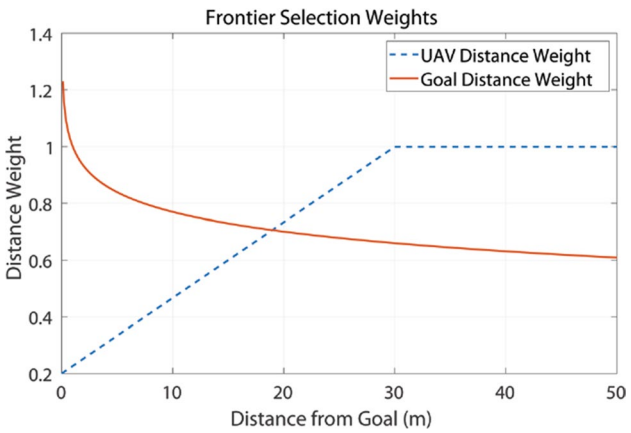

Fig. 13 Weights for the frontier selection cost function

Once the cost for all of the frontiers is calculated the one with the minimum cost is selected and the waypoint is sent to the navigation handler. This then directs the UAV to the selected frontier via a straight-line path.

6 Partial Plans for UGV

Allowing the UGV to move before a complete plan is found allows the overall mission time to be reduced. If the UGV waited at the starting location until the UAV was done exploring it would be guaranteed to not need to backtrack but would waste time that it could have been moving. Intermediate plans can be given to the UGV but they risk leading the UGV the wrong way and forcing it to backtrack. Thus, a balance between directly following the UAV and waiting at the start must be found. Figure 14 shows how the UGV can be sent partway to the frontier. The method proposed here uses a safe follow back distance along with looking for "danger areas" along the current path to the frontier.

The first part of the intermediate plans for the UGV is to generate a path to the frontier. Next, dangerous areas that could lead to backtracking need to be identified. Danger areas were defined as where obstacles abut with unknown space. These regions have high uncertainty and thus could lead to larger obstacles being uncovered. To quickly identify these regions the existing unknown and known area masks are used. The obstacle mask is inflated so it will overlap with areas in the unknown mask and a bit-wise and operation is done to quickly identify the danger regions.

Once the danger areas are identified and stored as a mask, the danger along the path needs to be quantified. The corridor on either side of the path is examined for grid cells marked as dangerous. The total danger at a point along the path is accumulated along a line perpendicular to the path. This is repeated for all points in the path to generate an array of danger scores along the path. Figure 15 shows the measured region on either side of the path.

To prevent the UGV from backtracking, it should be held back a certain distance from danger areas. A fixed distance from danger is defined as the follow back distance. Different follow back distances were tested in the simulation Section 7. These would be predetermined by operators based on the environment complexity. A sliding window with equivalent length of the follow back distance is applied to the path danger array starting at the end nearest the frontier. This window is slid back towards the start until the sum of the path danger in the window is less than a predefined threshold. The starting point of this window is then selected as the end waypoint for the UGV. The generated path up to this waypoint is sent to the UGV for execution.

7 Simulation Results

Having a simulation environment for the system is incredibly important as testing in the field is difficult for a variety of reasons. The software Gazebo handles the simulation of the environment, virtual cameras, and the vehicle physics. The UGV is simulated in Gazebo using the open source software provided by Clearpath Robotics. The UAV simulation is run using the Gazebo software in the loop (SITL) provided by the PX4 firmware developers. The only significant difference between the hardware and simulation environment is that the

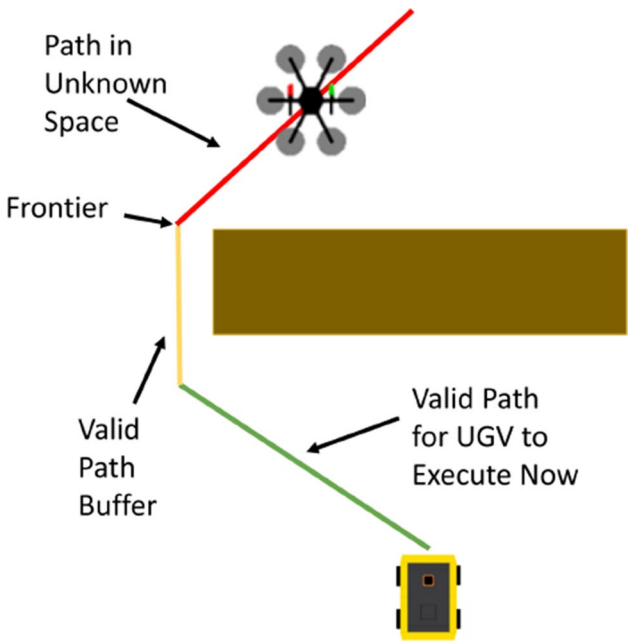

Fig. 14 Diagram showing how the UGV is held back when an incomplete plan to the goal exists

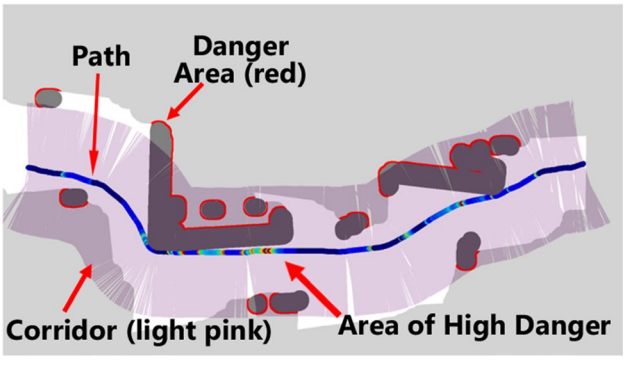

Fig. 15 UGV Path with the region around the path measured for danger areas. Increased danger along the path shows up on a color scale with the maximum being encoded as red

simulation uses a depth camera to simulate the stereo system on the real UAV.

7.1 Dijkstra Frontier vs RRT Explore Exploration Distance Comparisons

To compare the two algorithms, each were run on a variety of environments. Each environment consists of a different configuration of obstacles designed to challenge the algorithms. All environments used the same start and goal points with just different obstacles. The obstacles are constrained within an area 40 meters long and 10 meters wide with the start and goal on opposite ends of the 40m meter side. The size environment was selected based on the limited endurance of the Tarot 960 drone used for real-world testing. This size is about what the UAV can explore with its 15 minute flight time. A total of nine environments were created for evaluation but only three are presented in this section as they provide the most insight into the performance of the two algorithms. The environments not shown were either for testing other algorithms such as the UGV intermediate plan or ended up showing similar results to the environments already presented. Diagrams of each environment are shown in Appendix 1. The total distance traveled by the UAV finding a path for the UGV was recorded for 50 simulation runs for both the Dijkstra Frontier method and the RRT explore method.

The Demo 2 environment (Fig. 23) is designed to simulate an environment where an obstacle creates one path that is much better to explore than the other. This is done by offset obstacles that make the shortest path right through the center. The distances for this environment show that the Dijkstra Frontier method out performs the RRT explore method. This is expected as this was designed to be a worst case scenario for the RRT explore method. Since the RRT explore method is designed to reduce oscillations while exploring it commits to one side more than the Dijkstra frontier method which is always looking for the optimal path. This can be seen in the heatmaps (Fig. 17) and is plotted in Fig. 16. The heatmaps represents the frequency the UAV visited each part of the environment during exploration. The box plots show the summary statistics for all 50 simulations. The bottom and top edges of the box indicate the 25th and 75th percentiles, respectively. The Dijkstra Frontier method shows a pretty direct exploration through the center of the environment and never goes to the long sides of the obstacles. This is in stark contrast to the RRT explore method which over the course of the 50 simulations explored pretty much every path possible resulting in longer exploration distances (Fig. 17).

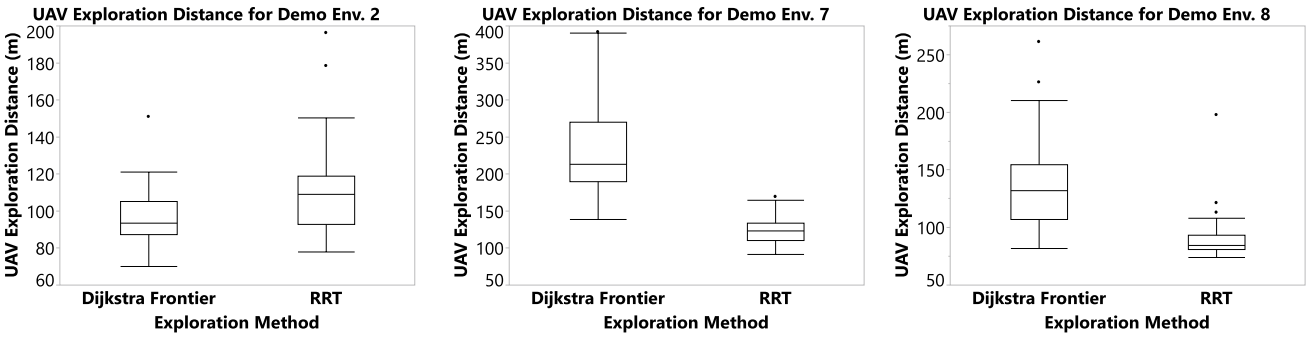

Fig. 16 Box and whisker plots of the total exploration distance of the UAV in three environments

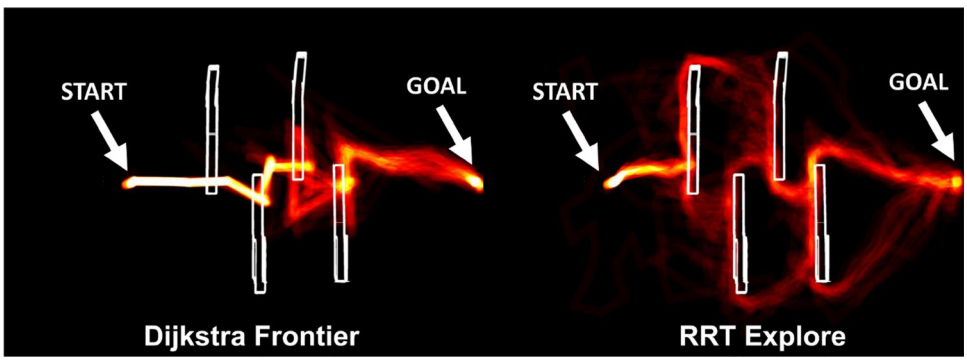

Fig. 17 Heatmap for Demo 2 environment, obstacles shown in white

The Demo 7 environment (Fig. 24) was setup to really test the advantage of the RRT explore method. The Dijkstra Frontier method tends to oscillate between the sides of a large obstacle, so two large obstacles were placed in between the start and the goal. The results (Fig. 16) show very clearly that the RRT explore method achieves shorter exploration distances with large obstacles in its way. The mean exploration distance was approximately 90 m longer for the Dijkstra Frontier method. This behavior can also be seen very clearly in the heatmaps (Fig. 18). The Dijkstra Frontier method shows exploration all over the environment while the RRT method shows fairly direct exploration around the perimeter of the obstacles.

The Demo 8 environment (Fig. 25) is designed to simulate a dense environment with a lot of small buildings in between the start and the goal. This was not designed to favor one algorithm but to simulate a realistic environment that the system could be asked to operate in. The results (Fig. 16) show that the RRT out preformed the Dijkstra Frontier method by around 40 m on average. Upon inspection of the heatmaps (Fig. 19) the main difference between the two is a tendency for the Dijkstra Frontier method to do more exploration around the obstacles similar as in the Demo 7 environment.

Overall the RRT explore algorithm shows the best average performance across a variety of environments. The Dijkstra Frontier method would likely be the better option in very simple environments where operators do not expect the vehicles to encounter many obstacles. In these simpler environments the Dijkstra Frontier method will produce the most direct/shorter route. For environments that the operators do not know what to expect or is known to have more complicated obstacles, the RRT explore method should be used.

7.2 UGV Intermediate Plan Method Results

The UGV intermediate plan method was tested in simulation with the follow back distance varied from 0 to 10 meters. As mentioned in section 6, the follow back distance is defined as a fixed distance from danger areas along the UGV's path. Thirty simulations were run for each distance. This was tested on an environment with (Fig. 25) and without a dead end (Fig. 26). The results from only these two environments, with and without a dead end, are presented as they represent the two extremes of environments this algorithm can expect. The dead end was around 15 vehicle lengths deep and 15 vehicle lengths wide. The results from both test environments are plotted in Fig. 20. Results for the no dead end environment showed increased mission times as expected and a slight decrease in UGV distance traveled. Results for the dead-end environment showed a slight increase in mission times but a significant decrease in UGV distance traveled. Overall the proposed method for controlling the UGV when the entire global plan is not yet known shows that holding the UGV back can result in distance traveled savings but not necessarily mission time savings. In environments with no dead ends, holding the UGV back only increases mission time. Only when there are possible dead ends should the UGV hold back distance be used. It should

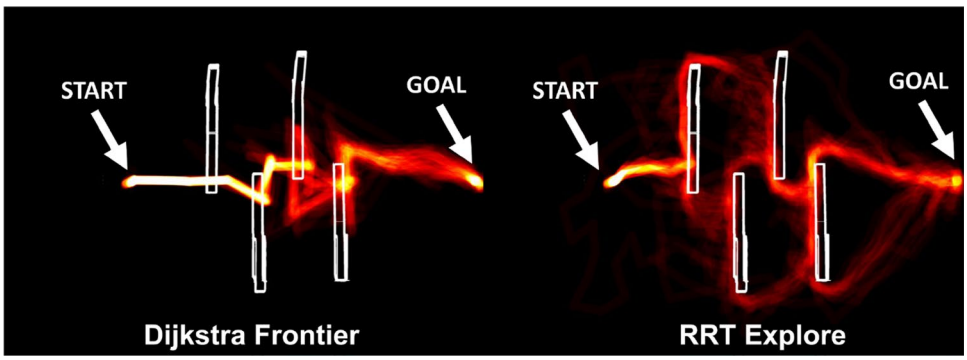

Fig. 18 Heatmap for Demo 7 environment, obstacles shown in white

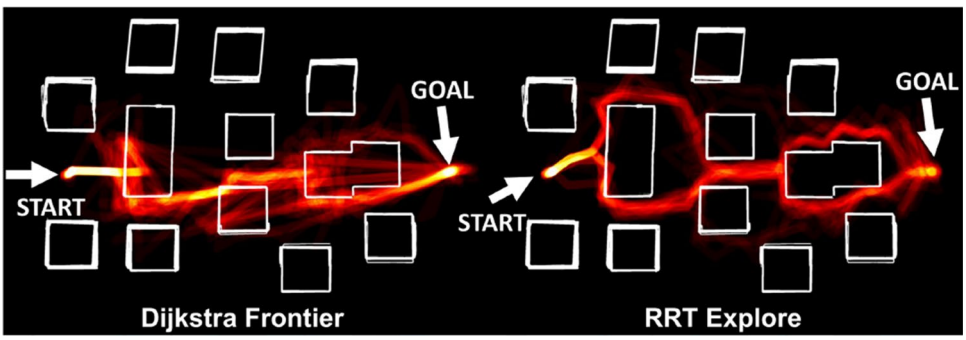

Fig. 19 Heatmap for Demo 8 environment, obstacles shown in white

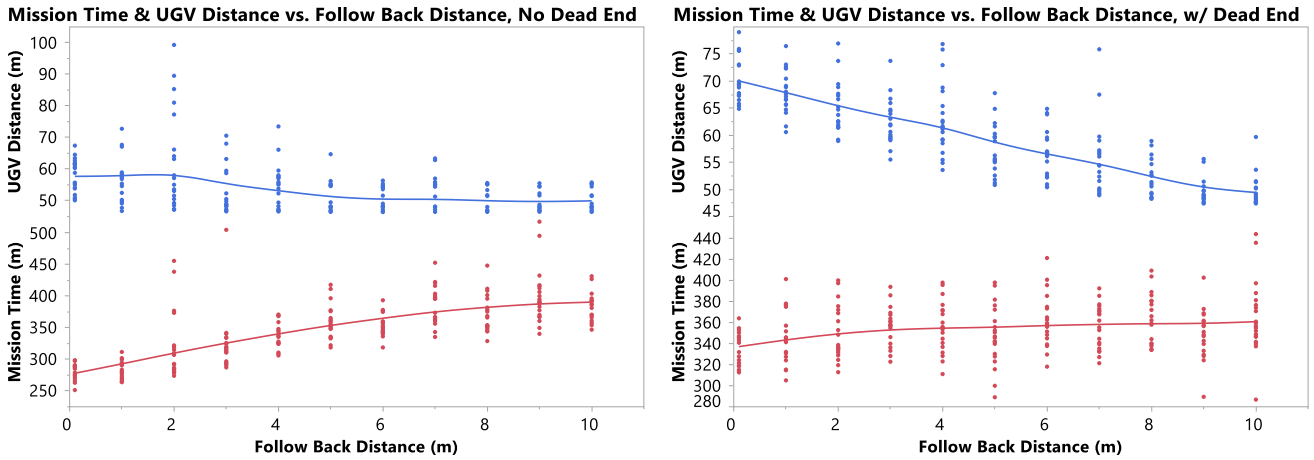

Fig. 20 Mission time and UGV distance for an environment without a dead end (left) and with (right)

be noted that mission time savings is also most likely a function of the UAV and UGV speed. If the UAV and UGV can move similar speeds then there will be no time savings from holding the UGV back. If the UGV is significantly slower there should be a point when holding the UGV back will result in time savings.

8 Field Demonstration

The system described here was tested at Virginia Tech's Experimental Aviation Systems Lab at Kentland Farms. The selected area has a good combination of buildings, grass, gravel, pavement, and vehicles that represent the type of environment the system was intended to operate in.

The first stage of the demonstration mission is to setup the vehicles, RTK GPS system, and operator control station. Next, the appropriate software is started on each of the three computers (UAV, UGV, OCS) and operation verified. The UAV pilot in command will then takeoff the UAV and fly it over to the RTK base station antenna for extrinsics calibration. The pilot then manually flies the UAV in the method described in Section 3 until enough samples are collected to calibrate the extrinsics of the stereo camera. Automating this step would be a good improvement to the system.

With the calibration complete the system is ready for the mission. The operator at the OCS will use either RVIZ or Mapviz to command a goal for the UGV and sends it to the UAV. The pilot will then switch the UAV into "Guided" mode which allows the software to control the UAV. The UAV will then begin running the RRT explore algorithm and start exploring the environment. As enough of the environment is explored the UGV will begin to follow behind using the method described in Section 6. Once the UAV explores the last part of the environment necessary for a complete plan for the UGV it will stop moving and the UGV will traverse to the goal. After the mission is complete, the operator can specify more goals for the UGV to go to and the process will repeat. The total distance/number of goals is mainly limited by the flight time of the UAV. After the operator is done sending goals to the UGV the pilot will take control of the UAV and land.

The demonstration at Kentland Farms consisted of one goal approximately $50m$ from where the UGV started on the other side of a barn. The environment, along with the vehicle starting locations, ending locations, and paths taken are shown in Fig. 21. Figure 22 shows the costmap generated by the system with areas of importance annotated. The mission was finished in approximately 7 minutes with the UAV traveling $\sim$ 100 meters from the UGV starting position and the UGV traveling $\sim$ 70 meters.

The computational time for the RRT exploration was recorded for both a desktop and the embedded TX2 on the UAV (see Table 1). The desktop tested has an AMD 1800X CPU, 32 GB of RAM, and a GTX 980Ti Graphics card. These values are the average for running the software on the recorded data from the field demonstration. The cycle time for the RRT was around 300 ms on board the TX2 which does not delay the UAV very much in flight.

The field demonstration was able to validate the improvements made to this UGV/UAV system. The onboard stereo mapping was shown to operate in real time and produce good maps for the UAV and UGV to plan on. This is a great improvement over the old system that only used a color-based obstacle detection technique. The long baseline stereo camera was able to produce good depth maps for obstacle detection. The exploration algorithm was shown to successfully generate a path for the UGV and provided a very direct route with the quicker UAV doing the exploration.

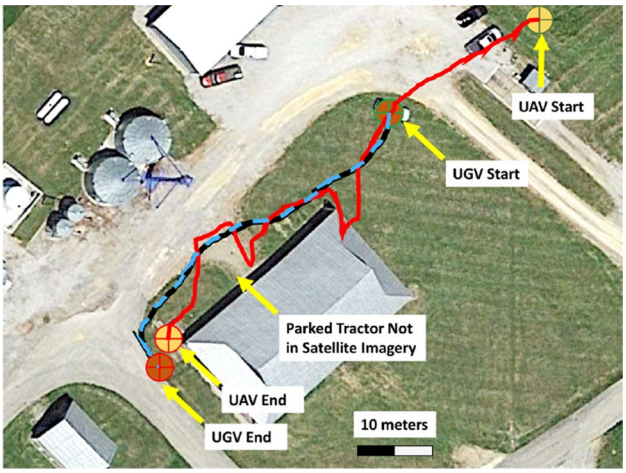

Fig. 21 Annotated map showing the paths of both vehicles

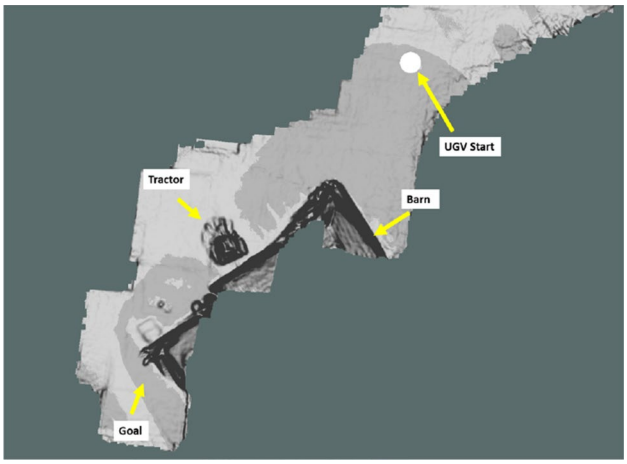

Fig. 22 Annotated costmap showing the start and goal position along with the barn and tractor seen during the mission

Table 1 RRT computational time per cycle

RRT explore Step	Desktop (ms)	NVIDIA TX2 (ms)
Mask Generation	24	66
Unknown RRT Generation	75	130
Known RRT Expansion	12	16
KD-Tree Creation	30	66
Connection and Frontier Extraction	11	25

Based on the field testing we identified several improvements for the system. The first is that the mission tempo could be greatly improved with using a UGV with a higher top speed. A large portion of the total mission cycle was the UAV hovering over the end waiting for the UGV to arrive. Another improvement would be to use the dynamic meshing feature in the Ubiquiti WIFI radios used. Dynamic meshing is the ability for the communication hardware to switch the path the signal goes using multiple radios. At the time of the experiment this was a beta feature and was not used. When the UGV went around the corner of the barn there was a significant drop in communication quality. Some other field testing has shown that the barn was enough of an obstacle to completely block communication if the UGV was on the complete opposite side. With the UAV hovering over the UGV on the other side and with a clear line of sight to the base station the mesh feature could reroute the communication to the UGV through the UAV.

9 Conclusion

This work presents the improvements to an existing novel cooperative unmanned aerial vehicle/unmanned ground vehicle (UAV/UGV) system. The cooperative UAV/UGV system aims to reduce risk for UGV exploration using the UAV to explore. Mapping was improved from the previous version of the system through adding a custom long-baseline stereo camera. The use of a stereo camera over a sensor, such as a lidar, greatly reduces the overall cost of such a system and fits within one of the goals of the project to have reasonable hardware requirements. The stereo computation was done onboard using the GPU. Mapping was also performed onboard with a new GPU based mapping implementation. Mapping registration to the world was improved using an online camera extrinsics calibration. Real-world projection errors of less than 0.5 meter were shown with this approach.

The presented RRT exploration algorithm directed the UAV to frontiers of interest allowing the UAV to explore the environment in an efficient manner to generate a path for the UGV to navigate to the specified goal. The RRT exploration algorithm showed good performance in both simulation and real-world environments. Furthermore, the algorithm was also shown to be efficient enough to run onboard the UAV on the NVIDA TX2 platform. The Dijkstra Frontier method was also presented and showed good performance in simple environments, but was outperformed in more complex, unstructured environments. A method to give the UGV partial plans before the environment was completely explored was presented. This method of holding back the UGV was shown to have potential distance traveled/energy savings for the UGV. The system was tested in simulation to verify its effectiveness and reliability. Finally, the system was tested on the hardware system in an outdoor scenario to demonstrate its capabilities. The outdoor scenario successfully demonstrated the effectiveness of the approach from both a hardware and software standpoint.

Since this is still an early developmental system there is plenty of opportunities for future research and development. The area that could benefit the most from further development

is the mapping subsystem. The main deficiency was the poor frame to frame registration between depth images projected onto the costmap. Poor registration limits the size of objects that can be successfully detected and can make traversable gaps in the environment appear impassable. Since only one outdoor scenario was tested, future work would also include testing the system in a wider variety of scenarios with different obstacle sets and degrees of difficulty. Other future work could include adding sensing the the UGV to improve its ability to navigate small scale obstacles in its local environment.

Appendix A: Simulation Environment Diagrams

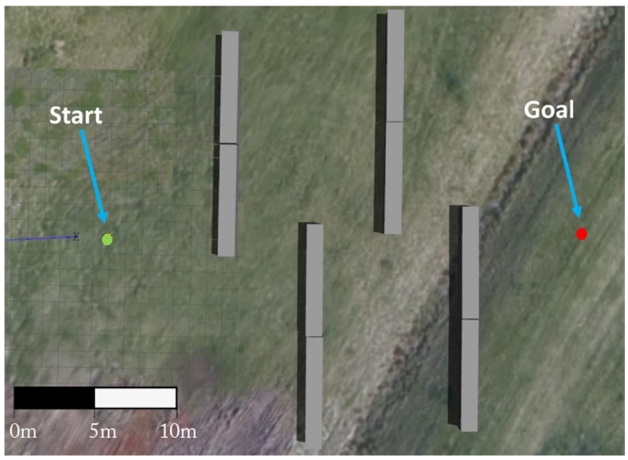

Fig. 23 Demo 2 environment. This environment features long obstacles with a much shorter path through the center. Designed to be a worst case for the RRT explore algorithm

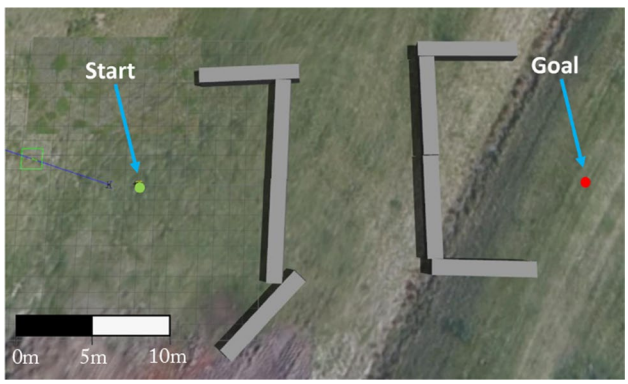

Fig. 24 Demo 7 environment. This environment features two C shaped obstacles one facing the start and one facing the goal.

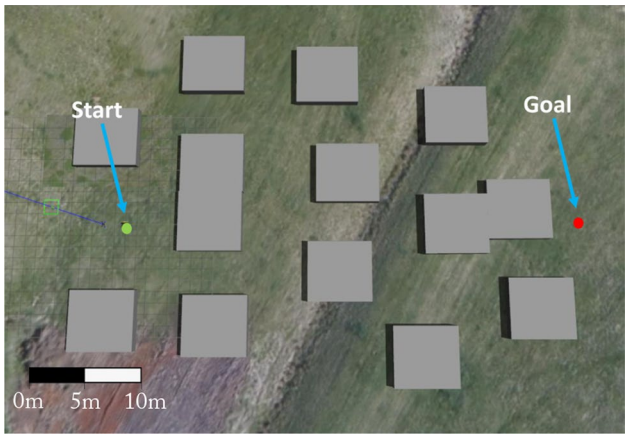

Fig. 25 Demo 8 environment. This environment features a large number of square obstacles to simulate an urban environment

Fig. 26 Demo 9 environment. This environment features a simple dead end near the start that is approximately 8 meters deep

Author Contributions KK, AW, JP, and JD conceived this research; AW, JP, JD, and SC contributed software to this research; AW, JP, JD, and SC performed experiments and analysis; AW, JP, and KK wrote the paper and participated in the revisions of it.

Funding This work was funded through the National Science Foundation I/UCRC Center for Unmanned Aircraft Systems Phase II Site Addition, Award No. 1650465.

Data Availability Data collected and used in this publication is available upon written request to Kevin Kochersberger at Virginia Tech

Code Availability The software referenced in this publication is available upon written request to Kevin Kochersberger at Virginia Tech

Declarations

Conflicts of Interest The authors declare that they have no conflict of interest.

References

1. Bachrach, A., He, R., Roy, N.: Autonomous flight in unknown indoor environments. Int. J. Micro Air Vehicles **1**(4), 217–228 (2009). https://doi.org/10.1260/175682909790291492
2. Bentley, J.L.: Multidimensional binary search trees used for associative searching. Communications of the ACM **18**(9), 509–517 (1975)
3. Burgard, W., Moors, M., Stachniss, C., Schneider, F.E.: Coordinated multi-robot exploration. IEEE Trans. Robot. **21**(3), 376–386 (2005). https://doi.org/10.1109/TRO.2004.839232
4. Cesare, K., Skeele, R., Hollinger, G.: Multi-uav exploration with limited communication and battery. In: 2015 IEEE International Conference on Robotics and Automation (ICRA), pp. 2230–2235 (2015). https://doi.org/10.1109/ICRA.2015.7139494
5. Davuluri, P., Chen, C.H.: Radio frequency interference due to usb3 connector radiation. In: 2013 IEEE International Symposium on Electromagnetic Compatibility, pp. 632–635. IEEE (2013)
6. Donnelly, J.J.: Semi-dense stereo reconstruction from aerial imagery for improved obstacle detection. Master's thesis, Virginia Tech (2019)
7. (FEMA), B.B.: Debris field in oklahoma (2008). https://commons.wikimedia.org/wiki/File:FEMA_-_35225_-_Debris_field_in_Oklahoma.jpg. Accessed 3 Jul 2019
8. Ferranti, E., Trigoni, N., Levene, M.: Brick mortar: an on-line multi-agent exploration algorithm. In: Proceedings 2007 IEEE International Conference on Robotics and Automation, pp. 761–767 (2007). https://doi.org/10.1109/ROBOT.2007.363078
9. González-Baños, H.H., Latombe, J.C.: Navigation strategies for exploring indoor environments. Int. J. Rob. Res. **21**(10–11), 829–848 (2002). https://doi.org/10.1177/0278364902021010834
10. Haubeck, K., Prinz, T.: A uav-based low-cost stereo camera system for archaeological surveys—experiences from doliche (turkey). Int. Arch. Photogramm. Remote Sens. Spat. Inf. Sci pp. 195–200 (2013)
11. Hood, S., Benson, K., Hamod, P., Madison, D., O'Kane, J.M., Rekleitis, I.: Bird's eye view: Cooperative exploration by ugv and uav. In: 2017 International Conference on Unmanned Aircraft Systems (ICUAS), pp. 247–255 (2017). https://doi.org/10.1109/ICUAS.2017.7991513
12. Juárez, D.H., Chacón, A., Espinosa, A., Vázquez, D., Moure, J.C., López, A.M.: Embedded real-time stereo estimation via semi-global matching on the GPU. In: International Conference on Computational Science 2016, ICCS 2016, 6-8 June 2016, San Diego, California, USA, Procedia Computer Science, vol. 80, pp. 143–153 (2016). https://doi.org/10.1016/j.procs.2016.05.305
13. Juliá, M., Gil, A., Reinoso, O.: A comparison of path planning strategies for autonomous exploration and mapping of unknown environments. Auton. Robot. **33**(4), 427–444 (2012). https://doi.org/10.1007/s10514-012-9298-8
14. Kalra, N., Ferguson, D., Stentz, A.: Hoplites: A market-based framework for planned tight coordination in multirobot teams. In: Proceedings of the 2005 IEEE International Conference on Robotics and Automation, pp. 1170–1177 (2005). https://doi.org/10.1109/ROBOT.2005.1570274
15. Kim, J.H., Kwon, J.W., Seo, J.: Multi-uav-based stereo vision system without gps for ground obstacle mapping to assist path planning of ugv. Electron. Lett. **50**(20), 1431–1432 (2014)
16. Kim, P., Price, L.C., Park, J., Cho, Y.K.: Uav-ugv cooperative 3d environmental mapping. In: ASCE International Conference on Computing in Civil Engineering 2019American Society of Civil Engineers (2019)
17. LaValle, S.M.: Planning Algorithms. Cambridge University Press, New York, NY, USA (2006)
18. Miki, T., Khrapchenkov, P., Hori, K.: Uav/ugv autonomous cooperation: Uav assists ugv to climb a cliff by attaching a tether. In: 2019 International Conference on Robotics and Automation (ICRA), pp. 8041–8047. IEEE (2019)
19. Milella, A., Reina, G., Foglia, M.M.: A multi-baseline stereo system for scene segmentation in natural environments. In: 2013 IEEE Conference on Technologies for Practical Robot Applications (TePRA), pp. 1–6. IEEE (2013)
20. Peterson, J., Chaudhry, H., Abdelatty, K., Bird, J., Kochersberger, K.: Online aerial terrain mapping for ground robot navigation. Sensors **18**(2), 630 (2018)
21. Rovira-Más, F., Wang, Q., Zhang, Q.: Design parameters for adjusting the visual field of binocular stereo cameras. Biosyst. Eng. **105**(1), 59–70 (2010)
22. Salas, W.L., Valentín-Coronado, L.M., Becerra, I., Ramírez-Pedraza, A.: Collaborative object search using heterogeneous mobile robots. In: 2021 IEEE International Autumn Meeting on Power, Electronics and Computing (ROPEC), vol. 5, pp. 1–6 (2021). https://doi.org/10.1109/ROPEC53248.2021.9668148
23. Schneider, J., Eling, C., Klingbeil, L., Kuhlmann, H., Förstner, W., Stachniss, C.: Fast and effective online pose estimation and mapping for uavs. In: 2016 IEEE International Conference on Robotics and Automation (ICRA), pp. 4784–4791. IEEE (2016)
24. Shim, D., Chung, H., Kim, H.J., Sastry, S.: Autonomous exploration in unknown urban environments for unmanned aerial vehicles. IEEE Robotics & Automation Magazine **13** (2005). https://doi.org/10.2514/6.2005-6478
25. Umari, H., Mukhopadhyay, S.: Autonomous robotic exploration based on multiple rapidly-exploring randomized trees. In: 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 1396–1402 (2017). https://doi.org/10.1109/IROS.2017.8202319
26. Warren, M., McKinnon, D., Upcroft, B.: Online calibration of stereo rigs for long-term autonomy. In: 2013 IEEE International Conference on Robotics and Automation, pp. 3692–3698. IEEE (2013)
27. Yamauchi, B.: A frontier-based approach for autonomous exploration. In: In Proceedings of the IEEE International Symposium on Computational Intelligence, Robotics and Automation, pp. 146–151 (1997)

Anthony Wagner received his Master's degree in mechanical engineering from Virginia Tech, Blacksburg, Virginia in 2019. His research interests are autonomous aerial systems, path planning, and collaborative systems.

John Peterson received his Ph.D. in mechanical engineering from Virginia Tech, Blacksburg, Virginia in 2020. His research interests are autonomous aerial systems, unmanned ground vehicles, and radiation mapping.

James Donnelly received his Master's degree in mechanical engineering from Virginia Tech, Blacksburg, Virginia in 2019. His research interests are autonomous aerial systems, stereo vision, and 3D scene reconstruction & mapping.

Shivam Chourey received his Master's degree in computer engineering from Virginia Tech, Blacksburg, Virginia in 2020. His research interests are autonomous aerial systems, path planning, and human-robot cooperation.

Kevin Kochersberger received his Ph.D. in mechanical engineering from Virginia Tech, Blacksburg, Virginia in 1994. He is currently an Associate Professor at Virginia Tech. His research interests are autonomous aerial systems, dynamics and control, and applied aerodynamics.