Autonomous Exploration and Motion Planning for an Unmanned Aerial Vehicle Navigating Rivers

Stephen Nuske, Sanjiban Choudhury, Sezal Jain, Andrew Chambers, Luke Yoder, and Sebastian Scherer
Robotics Institute, Carnegie Mellon University, Pittsburgh, Pennsylvania 15213
Lyle Chamberlain and Hugh Cover
Near Earth Autonomy, Inc., Pittsburgh, Pennsylvania 15237-1523
Sanjiv Singh
Robotics Institute, Carnegie Mellon University and Near Earth Autonomy, Inc., Pittsburgh, Pennsylvania 15237-1523

Received 12 July 2014; accepted 16 February 2015

Mapping a river's geometry provides valuable information to help understand the topology and health of an environment and deduce other attributes such as which types of surface vessels could traverse the river. While many rivers can be mapped from satellite imagery, smaller rivers that pass through dense vegetation are occluded. We develop a micro air vehicle (MAV) that operates beneath the tree line, detects and maps the river, and plans paths around three-dimensional (3D) obstacles (such as overhanging tree branches) to navigate rivers purely with onboard sensing, with no GPS and no prior map. We present the two enabling algorithms for exploration and for 3D motion planning. We extract high-level goal-points using a novel exploration algorithm that uses multiple layers of information to maximize the length of the river that is explored during a mission. We also present an efficient modification to the SPARTAN (Sparse Tangential Network) algorithm called SPARTAN-lite, which exploits geodesic properties on smooth manifolds of a tangential surface around obstacles to plan rapidly through free space. Using limited onboard resources, the exploration and planning algorithms together compute trajectories through complex unstructured and unknown terrain, a capability rarely demonstrated by flying vehicles operating over rivers or over ground. We evaluate our approach against commonly employed algorithms and compare guidance decisions made by our system to those made by a human piloting a boat carrying our system over multiple kilometers. We also present fully autonomous flights on riverine environments generating 3D maps over several hundred-meter stretches of tight winding rivers.

1. INTRODUCTION

Riverine applications are an increasingly important focus for many tasks, such as mapping, monitoring, and surveillance, for which it is desirable to use autonomous exploration to traverse the river and collect up-to-date information. A small lightweight system that can travel below the tree line to sense the river width, the river direction, and canopy clearance is advantageous since this information often cannot be measured from satellite imagery because tree canopy cover occludes the river from above. Furthermore, narrow densely forested rivers are difficult to navigate by surface craft because of submerged and semisubmerged obstacles; therefore, we develop a micro aerial vehicle (MAV) that is small and nimble and can traverse the difficult terrain.

*Near Earth Autonomy Inc., info@nearearth.aero, www.nearearth.us
Direct correspondence to Stephen Nuske: nuske@cmu.edu

Existing applications for autonomous river operations focus on collecting information from robotic boats navigating based on stored directions. Existing platforms have used predetermined GPS waypoints as a navigation guide. Often riverine systems are densely overgrown with vegetation, and autonomous exploration cannot depend on the irregular and erroneous GPS measurements in these surroundings. Due to their dense canopy, estimating an initial map of the waterways from satellite images is also not a viable option. Furthermore, riverine environments are also continuously evolving, and current information on the width and course is often not available. For all of these reasons, riverine environments must be explored without relying on predetermined maps or waypoints. To date, there have been few, if any, truly autonomous exploration systems demonstrated in a riverine environment. We present what we believe is the first such system that can explore a river solely from local sensing, and our vehicle is the first that can fly through river environments to quickly explore while avoiding submerged and semisubmerged obstacles.

View this article online at wileyonlinelibrary.com • DOI: 10.1002/rob.21596

This paper builds on our existing work in river environments for autonomous perception, positioning, and obstacle avoidance work (Achar, Sankaran, Nuske, Scherer, & Singh, 2011; Chambers et al., 2011; Cover, Choudhury, Scherer, & Singh, 2013; Rehder, Gupta, Nuske, & Singh, 2012; Scherer et al., 2012) and extends to contribute the key capabilities of the following:

1. A modified low-level motion planning algorithm called SPARTAN-lite (where SPARTAN denotes Sparse Tangential Network) that exploits geodesic properties on smooth tangential manifolds around obstacles.
2. A novel exploration algorithm [originally presented in Jain et al. (2013)] that enables truly autonomous exploration of rivers using a multivariate cost map.

The motion-planning algorithm, called SPARTAN-lite, enables autonomous flight in the unstructured riverine environment. The task intrinsically demands consistently delivering safe, feasible paths in real time. Such environments tend to have clusters of cluttered objects separated by large segments of empty space. Standard motion-planning approaches distribute computational effort evenly between free space and obstacle-dense regions. There is a need for a lightweight planner that computes only when it needs to and sits idle otherwise. We present a motion planner that leverages the intermittent sparsity of its environment and creates plans several times faster than the state-of-the-art (Chambers et al., 2011). The planner constructs a graph wrapped in a tangential surface around obstacles. The large speedups have been achieved by creating vertices during the obstacle cost update process as well as by creating careful graph construction rules to remove redundancies. Using limited onboard resources, SPARTAN has flown autonomously along rivers, around trees, under bridges and power lines, and over changing terrain. Here we present an updated version of SPARTAN, called SPARTAN-lite, which exploits smooth manifolds around obstacles.

The exploration algorithm provides high-level goal points for the motion-planning algorithm. We use two sensor modalities: a three-dimensional (3D) spinning laser scanner and an RGB stereo camera pair. For the laser scanner, we develop an approach to identify the river., that models both laser beams that return as specular reflections from the river surface and those that are scattered and absorbed by the river. For the camera, we use a self-supervised river segmentation method that we presented in a prior publication (Achar et al., 2011; Scherer et al., 2012). Both sensors provide a map of the river extent and the bank of the river, which we use as input to the exploration algorithm. From the river and bank environment model, we extract three forms of information relating to free-space, time since observation, and distance of observation from vehicle. These combine into a fused cost map from which a goal-point is robustly extracted. We demonstrate that our method is more adept than traditional exploration algorithms that often focus solely on detecting the frontier between free-space and unknown areas. Our approach demonstrates an increased amount of information of the river gathered during a mission. The output of the system is a map of the course and width of a river, internally when evaluating the exploration algorithm, and we define a metric of success as increasing the length of the riverbank observed.

Results are collected from both simulation and real-world experiments. The simulations evaluate the algorithms against implementations of other commonly deployed approaches. The real-world experiments include a boat manually driven down a river with the sensor mounted onboard and also fully autonomous flights with the micro unmanned aerial vehicle (UAV) navigating several hundred meters of lakes and rivers without GPS and without any prior map (see a video of the autonomous flight at https://www.youtube.com/watch?v=vaKNbzYSK6U).

2. RELATED WORK

Much work has been put into the development of autonomous vehicles for navigating waterways, using a variety of different types of craft such as automated catamarans (Dunbabin, Grinham, & Udy, 2009; Pradalier, Posch, Pernthaler, & Siegwart, 2012), small lightweight fanboats (Valada et al., 2012), kayaks (Leedekerken, Fallon, & Leonard, 2010), or small inflatable craft (Gadre, Du, & Stilwell, 2012). Existing waterway navigation systems rely on predefined GPS-waypoints (Valada et al., 2012), or a predefined map generated from satellite imagery (Gadre et al., 2012). In contrast, our work is focused on autonomous exploration, where the environment is perceived by onboard sensors and the vehicle reacts by planning routes that navigate the vehicle along the waterway and mapping the direction and width of the riverbank.

We achieve this with a spinning 3D laser scanner, which has also been demonstrated for local obstacle avoidance (Dunbabin et al., 2009; Gadre et al., 2012) to navigate around obstacles discovered above the water surface. However, we do not use any prior information, and we rely on intelligent path and goal planning based on the information received by our local sensing. In one somewhat related work (Rathinam et al., 2007), rivers are detected and tracked from aerial vehicles, although unlike our work these are higher-flying vehicles, making them as unsuitable as satellite images, whereas our system operates beneath the tree-line.

In terms of exploration strategies, a common approach is to optimize the robot pose accuracy and the accuracy of the resulting map (Amigoni & Caglioti, 2010; Kollar & Roy, 2008). In contrast, we rely on separate positioning algorithms (Rehder et al., 2012) for pose accuracy, and we focus our exploration algorithm to maximize the length of riverbank discovered. In some exploration strategies, information maximization is focused on reducing uncertainty in

the pose and likelihood of map cells being an obstacle or free space (Amigoni & Caglioti, 2010; Kollar & Roy, 2008). Other approaches more closely related to ours define exploration goals that select viewpoints that are expected to yield the highest increase in entropy (Moorehead, Simmons, & Whittaker, 2001; Stachniss & Burgard, 2003) resulting in the robot seeking out regions that have not yet been explored. Overall, these strategies are closely related to the standard frontier exploration systems (Yamauchi, 1997). Our method is similar in nature, although we introduce a multivariate cost-map, which finds trajectories that maximize the length of a river explored for a given mission time by specifically trying to increase the amount of the riverbank observed.

The survey by Kendoul (2012) classifies most practical planning approaches for UAVs in outdoor environments, while Goerzen, Kong, & Mettler (2010) make a more broad classification. SPARTAN falls in the category of generating a roadmap for free space using a representation such as a visibility-graph and searching over this roadmap.

The most popular approach to planning in 3D spaces is by performing a Heuristic Graph Search as summarized by Ferguson, Likhachev, & Stentz (2005). This search can be performed over a generic roadmap or over regularly connected grids. Even though the latter is more popular because the graph is easier to construct, it becomes very computationally heavy for fine resolutions (submeter). Multiresolution methods have effectively overcome this problem. For example, unmanned helicopters have been flown by Tsenkov et al. (2008) and Whalley, Tsenkov, Takahashi, Schulein, & Goerzen (2009) using a quad-tree grid representation and performing an A* search. However, such methods still incur discretization errors and produce unnatural oblique paths.

Probabilistic methods allow solutions of arbitrary complexity and resolution and probabilistically converge on a solution. By sampling uniformly in free space, probabilistic roadmaps (Kavraki, Svestka, Latombe, & Overmars, 1996) can be generated for graph search, or rapidly exploring random trees (LaValle, 1998) for faster solutions. However, the convergence of these methods is slow, the plans found are not optimal, and no performance bounds can be claimed about them. More recently, the introduction of the Rapidly-exploring Random Trees – Star (RRT*) by Karaman & Frazzoli (2010) improves the performance by bringing asymptotic optimality to sampling-based planners.

Visibility graphs and their reduced forms have been analyzed in detail by LaValle (2006). Despite the computational complexity, it has been used for navigation in an outdoor scenario by Wooden & Egerstedt (2006). In UAV planning, visibility graphs have been used by Hoffmann, Waslander, & Tomlin (2008a) to generate a 2D initial guess to navigate within a polygonal environment. It also has been used to plan in a limited horizon by Kuwata & How (2004). However, these methods project real data into a polygonal environment and plan in a geometric space, thus making the method sensitive to sensor noise. Similar in nature to visibility methods, the tactic of staying close to an obstacle and moving tangentially to it is present in reactive planning techniques (Hrabar, 2011). This algorithm checks collisions within a cylindrical safety volume, and finds an escape point. The 3D Dodger approach by Scherer, Singh, Chamberlain, & Elgersma (2008) also results in trajectories lying on a tangent from obstacles.

2.1. System Overview

Earlier incarnations of our system were described in Chambers et al. (2011) and Scherer et al. (2012). These were partial descriptions, and in some cases the descriptions now do not reflect the current state of the system. In Figure 2, we present the current details of the system implementation.

A picture of our system can be seen in Figure 1(b). It is an eight-rotor, battery-powered micro UAV. The base platform is a Mikrokoptor bought off-the-shelf, and, retrofit with our own sensors and computing payload and interface to provide autonomous commands.

Sensor Suite: Figure 1(b) highlights the two main sensors, namely a color camera and a custom-built 3D spinning laser range scanner. We also carry our own inertial measurement unit (IMU) and barometric sensor to help estimate the pose of the vehicle.

We mount the sensors on a lightweight, carbon-fiber mount, which we custom-build. The carbon-fiber mount is attached via shock isolating gel mounts to reduce vibrations from the main platform.

Stereo-Camera: For six-degrees-of-freedom (6-DOF) relative motion and for detecting the river extent, we use a stereo-camera that is comprised of two IDS UI-1240SE-C-HQ cameras with Lensagon BM4018S118 4 mm 1/1.8 in. format wide field-of-view lenses. The stereo-pair is mounted on the lightweight carbon-fiber mount at about a 400 mm baseline.

Custom 3D Spinning Laser Scanner: For robust and accurate obstacle sensing, we have developed a 3D spinning mechanism for a 2D laser scanner. The scanner is a 2D Hokuyo UTM-30LX-EW that is actuated to provide a 3D scan pattern using a custom-built rotating mechanism. The scanner is described in detail in Chambers et al. (2011) and Scherer et al. (2012).

IMU: We use a Microstrain 3DM-GX3-45 integrated IMU/GPS unit, although for these experiments we explicitly do not use the GPS measurements and operate completely GPS-denied. We do use the IMU measurements for high-frequency, low latency positioning updates for our state filter.

Stereo Visual Odometry: We implement a version of Libviso2 (Geiger, Ziegler, & Stiller, 2011) with custom image matching software implemented and distributed on a custom processor board with four Logic PD Torpedo Computer on Modules (COMs). Each COM has a 1 GHz ARM Cortex-A8 processor with a TMS320C64x+ DSP

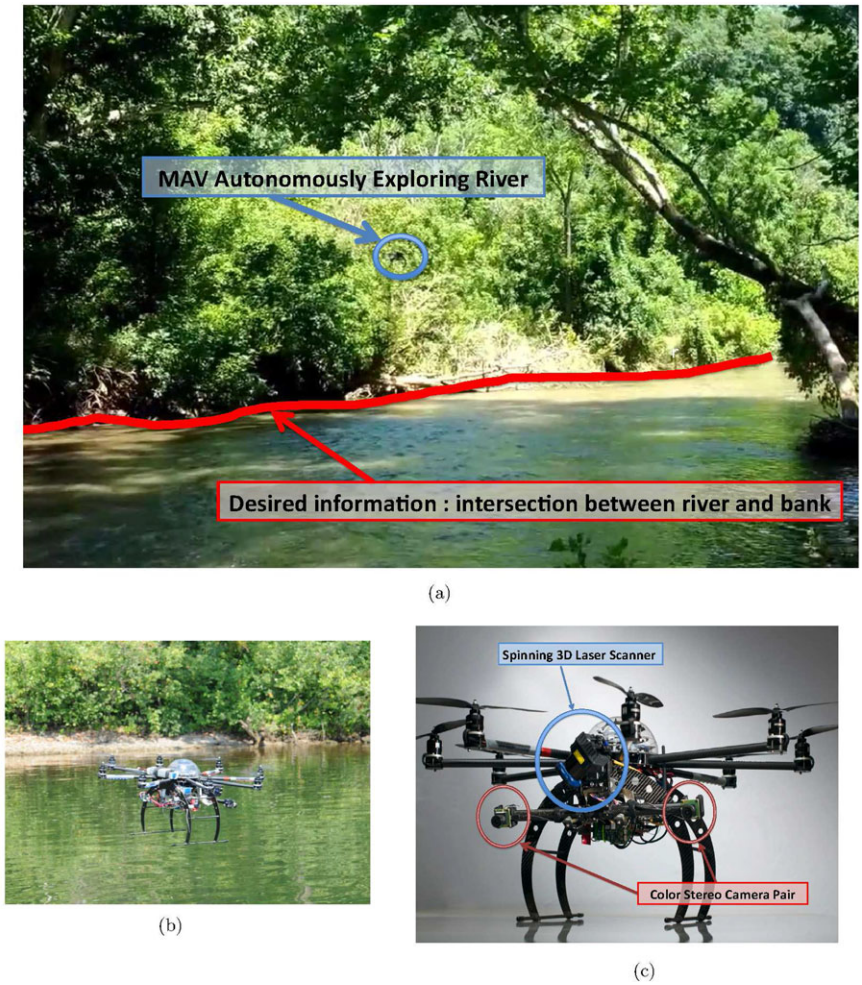


Figure 1. A micro aerial vehicle is used to autonomously explore and map the river environment. The information of interest is the intersection between bank and river. The vehicle is lightweight and agile and not susceptible to submerged and semisubmerged obstacles such as would be hazardous to an aquatic surface vessel. To avoid obstacles and to perceive the extent and course of the river, the vehicle is fitted with a spinning 3D laser scanner and RGB stereo camera pair.

coprocessor. We have also developed an algorithm to correct for bias in visual odometry occurring from tracking distant features (Rehder et al., 2012).

Height Above River Estimation: We maintain a coordinate frame relative to the river, and to keep track of the height above the river we use specular laser returns directly beneath the vehicle.

Barometric: For situations in which the vehicle is not above a river, we use an onboard barometric sensor (Weather Board - USB, SEN-10586, sparkfun.com) for measuring altitude changes.

State Estimation (UKF): We fuse stereo visual odometry, IMU measurements, and altitude measurements with an unscented Kalman filter. This approach is presented in detail in Chambers et al. (2014).

3D Obstacle Mapping: We use efficient algorithms for taking 3D spinning laser data and producing online 3D obstacle maps that are used for online motion planning; details are presented in Scherer et al. (2012).

2D River Mapping: We use camera and lidar data to generate river maps to be used by the exploration algorithm. The initial version of the approach to generate the river maps was presented in Chambers et al. (2011), and current details are presented in Section 3.1.

Exploration Algorithm: To generate goal points to enable the vehicle to explore the river, we have developed an exploration algorithm, presented in Section 3.

Motion Planning: The goals for the vehicle are then passed to a local motion planner that efficiently generates

3D trajectories for the vehicle, described later in Sections 4, 5, and 6.

Flight Control: The trajectories are provided to a trajectory controller that minimizes cross-track and along-track error to produce a desired attitude and acceleration.

This paper proceeds in the following sections to describe the exploration and motion planning algorithms in detail.

3. EXPLORATION ALGORITHM

The key aim of our exploration algorithm [originally presented in Jain et al. (2013)] is to find goal points to pass to the motion planning algorithm (SPARTAN-lite—Section 6) for the vehicle to execute trajectories to realize the following behaviors:

- Follow river, while maintaining stable flight and avoiding obstacles.
- Maximize the information collected over the course of the river.

3.1. Environment Modeling and Sensing for Goal Planning

The exploration algorithm needs to consider the river extent to derive goals that maximize the length of the river that is explored. We specifically decide to consider the amount of the riverbank explored. The alternative would be also to try to maximize the river width observed as well, but we have chosen not to because when we are in wide rivers we do not want to zigzag back and forth across the river, thus we prefer to navigate as far down wide rivers as possible. Here, the riverine system is modeled as a planar grid (χ). Each cell χ^i in the grid represents a cell in the world at location x_i, y_i, and the rivermap values of this cell in the grid are as follows:

$$\chi_r^i = \chi_r^{x_i, y_i} = \begin{cases} 1 & \text{if the cell is part of the river,} \\ -1 & \text{if the cell is part of the bank,} \\ 0 & \text{if the cell has not been observed.} \end{cases} \quad (1)$$

Taking χ, we form a function that defines the current information that we have about the river. We define the intersection between river and bank as the pertinent information for our algorithm, and we use these cells as a measure of information. To achieve this, we form a new information map as follows to search for discontinuities in the current river model:

$$I^{(x_i, y_i)} = \sum_{u=x_i-1}^{u=x_i+1} \sum_{v=y_i-1}^{v=y_i+1} [\mathrm{sgn}(\chi_r^{(x_i, y_i)}) \neq \mathrm{sgn}(\chi_r^{(u,v)})]. \quad (2)$$

Our exploration algorithm seeks to extract desirable trajectories for the vehicle that will maximize the entropy in I.

3.2. Laser Classification

The above formulation is derived from data collected from local sensing mounted onboard the vehicle; see Figure 1(c). The laser scanner and the camera onboard generate environment maps that are used for goal planning.

We use a lightweight spinning laser range scanner to create a 3D scan of the environment [see Figures 1(c) and 3 and Scherer et al. (2012)]. The maximum range of this scanner depends on ambient illumination and reflectance characteristics of objects, but in typical outdoor environments we observe maximum ranges of approximately 15 m. We use this range to determine which laser missed returns are due to limited range and which are due to water absorption, i.e., we can detect the river from these laser misses. The laser range measurements are converted into a 3D point cloud in the world frame using an accurate positioning system that operates in GPS-denied environments (Rehder et al., 2012; Scherer et al., 2012). We measure the current height above the river surface, which cannot be derived purely from the global frame, since the height will vary according to the current water level. To achieve this, we extract specular returns from the water surface in a tight cone directly below the vehicle.

Once the global position and relative height above the surface are known, we can then proceed to use the laser measurements to form our environment map. In particular, the following rules are applied:

- All *missed* laser returns (those with the maximum laser range) that pass through the river plane are considered as river cells at the intersection of the ray and the river plane $\{\chi_r^i = 1\}$.
- All laser hits less than maximum range are projected on the environment grid, and based on the density of these projected hits in a cell, the cell is classified as part of the river bank $\{\chi_r^i = -1\}$.

3.3. Visual River/Bank Classification

Classification of the environment into river and bank through vision is done by performing self-supervised riverbank segmentation on the image captured by the camera onboard the vehicle (Achar et al., 2011).

Figure 4(b) illustrates the precision rates of a cell being classified as river/bank by comparing the segmentation output to an offline map of the environment. The visual classification is less reliable short ranges in comparison to the laser scanner. However, the range of the camera is much greater than the laser, and with a precision of river classification it is 80%, which is satisfactory to enable longer-range exploration decisions to be made. The performances of the visual and laser scanner river sensing approaches are evaluated against each other against a human operator in the results section (Fig. 14).

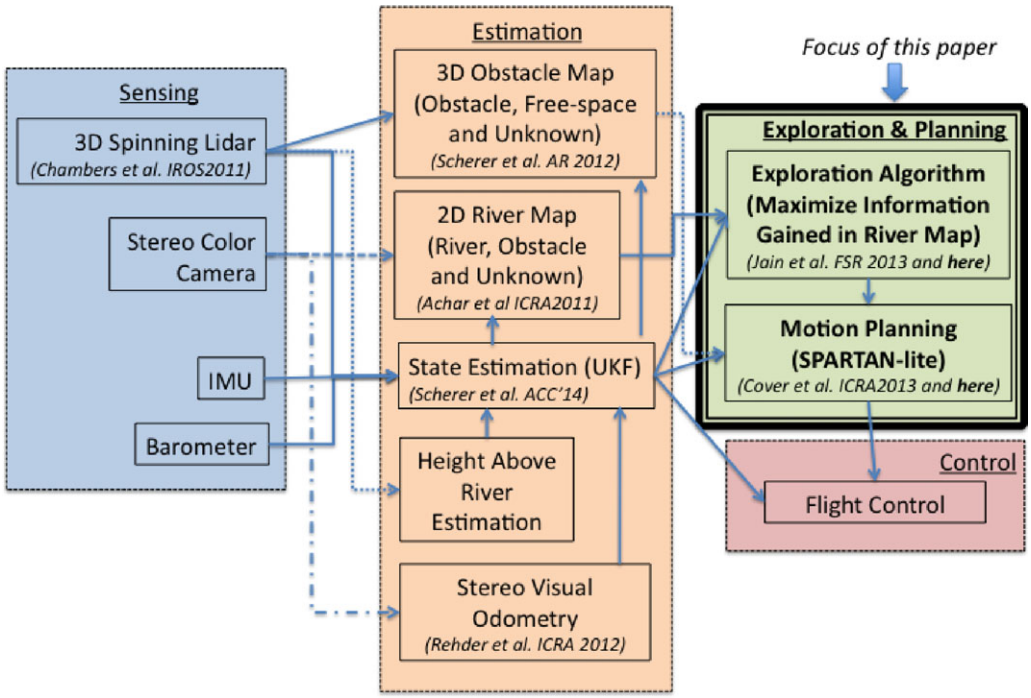


Figure 2. Overview of the entire autonomous system. In this paper, we focus on the exploration and planning algorithms; here we seek to highlight how these algorithms fit into the system at large.

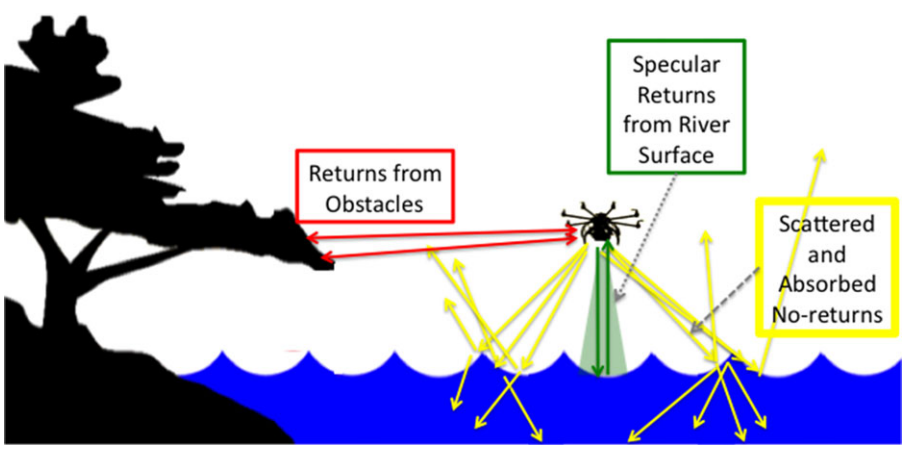
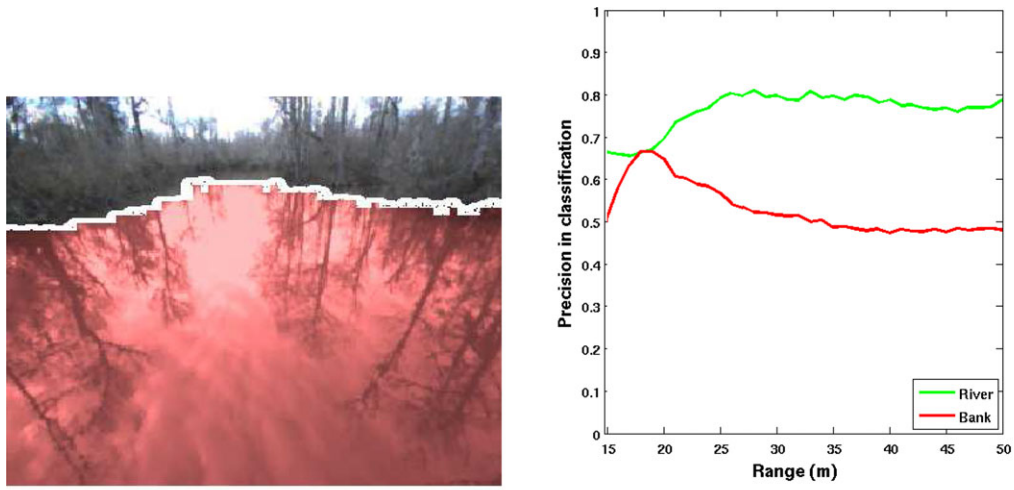


Figure 3. Diagram depicting a spinning laser scanner used for detecting the river, estimating the height above the river, and detecting obstacles on the riverbank. Laser returns are recorded from the water surface in a cone directly beneath the vehicle from specular reflections, which can be used to estimate the height above the river. Few, or no, laser returns are recorded from the river outside of a cone directly beneath the vehicle since the laser scatters and absorbs into the water surface. We infer the presence of the river from this lack of returns. Laser returns recorded from obstacles within range on the riverbank (approximately 15 m for Hokuyo in an outdoor environment).

3.4. Goal-point Extraction for River Exploration

The main task in autonomous exploration is to take a local perception of the environment and to extract goals for the vehicle to traverse toward. The goals are then fed in as input to the low-level motion planning algorithm presented in Section 5. The exploration algorithm we present here sets goals that seek to *maximize the information gained during the mission*.



(a) (b)

Figure 4. Left: example of a self-supervised segmentation algorithm detecting the river within camera images. Pixels with a probability greater than 0.5 after river segmentation are classified as river and marked with a white contour (Achar et al., 2011). They are then projected onto the river plane to create an initial map of the environment. This image is from a dataset taken from the McCarthy river, Mississippi. Right: visual River/Bank Classification. These results are collected from a 1-km-long run on a winding river. The precision rate of approximately 80% at long distance from the vehicle indicates the usefulness of visual classification for making long-range exploration decisions.

To achieve the desired behaviors, we introduce multivariate cost maps that respect the characteristics of the sensing and extend the abilities of more simplistic traditional frontier exploration algorithms (Yamauchi, 1997). In particular, the costs we derive enable the vehicle to observe the maximum length of the riverbank while following the course of the river, and where possible avoid returning to unexplored portions of the river that are behind the vehicle. Frontier that is not observed as the vehicle passes by initially may be larger in size than a narrow passage that the vehicle encounters directly ahead, however it is suboptimal to return to these locations as little new information is collected on the journey back to previously explored areas.

We develop a riverbank-hugging behavior that uses a distance transform-based cost function, $C_d(\cdot)$, that aims to keep the vehicle *away from* but *near enough* to the riverbank to both assist the 3D mapping of the bank and to ensure the local motion estimation is functional. This range is designed to result in maximal information gain of the riverbank. To arrive at $C_D(i)$, we compute a distance transform $f_D(i)$ that returns the distance to the nearest obstacle ($\chi_r^j < 0$). We efficiently compute this distance cost as described in detail (including timing information) in Scherer et al. (2012). After calculating the distance transform, we apply a function to penalize goals very close to obstacles, and we also penalize goals far away from obstacles using a desired distance κ_D as follows:

$$C_D(i) = 1 - \exp\left(k_D[f_D(i) - \kappa_D]^2 + \frac{1}{f_D(i)}\right), \tag{3}$$

$$f_D(i) = \underset{\substack{j=1:N \\ \chi_r^j < 0}}{\operatorname{argmin}} ||(x_i, y_i) - (x_j, y_j)||, \tag{4}$$

where k_d is a tuning constant. The resulting functional is depicted in Figure 5(a), where the cost is high near the obstacles and descends to a minima at κ_D. The behavior is to keep following clear passages in the river ahead, keeping a safe distance from the bank, and generally only in the case in which an obstacle blocks the entire river will the vehicle come to a stop.

The next cost we introduce is designed to avoid retracing steps. In particular, we assign cells that have been observed more recently with lower cost than those behind the vehicle, which were observed further in the past. We take the elapsed time since the ith cell was last observed as χ_t^i, and we use it to penalize retracing through cells seen previously as follows:

$$C_T(i) = t - \chi_t^i. \tag{5}$$

Figure 5(b) visualizes this temporal observation cost. An important cost we introduce is $C_R(i)$. The range the cell is from the current vehicle-location, which is designed to maximize the distance traversed along the river:

$$C_R(i) = ||(x_i, y_i) - (x_t, y_t)||, \tag{6}$$

where (x_t, y_t) is the current position of the vehicle; see Figure 5(c).

Next we introduce a cost to favor the vehicle continuing on its current course to avoid the issue of isotropic sensor

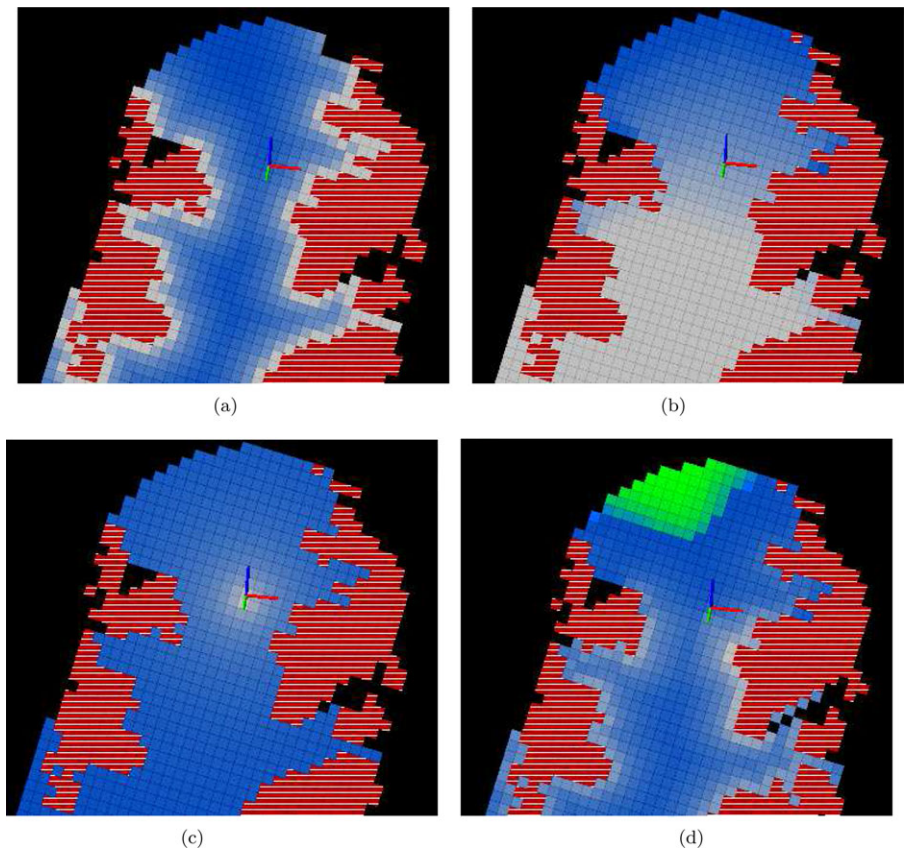


Figure 5. Visualizations of the multivariate cost functions. Obstacles are highlighted by lines, and the cells observed as the river are rendered a shade from white to blue (dark), where deep blue represents low cost to go. Cells with lowest cost in a map represent the next goal point for navigation. For the combined cost functions, we indicate the cluster of lowest cost cells in green in (d).

input that typically occurs at the commencement of a mission when no obstacles are within range, and the aforementioned costs are at an equilibrium and do not return stable goals,

$$C_H(i) = \exp^{\kappa_H(\theta_z^v - \Delta_\theta)}, \tag{7}$$

$$\Delta_\theta = \arctan\left[\left(\frac{x_t - x_i}{y_t - y_i}\right)^2\right], \tag{8}$$

where k_H and κ_H are constants that are empirically determined to create a dip in cost around zero heading in body-frame coordinates, to enable the vehicle to maintain its course when the sensory inputs do not provide stable goals, such as in open water.

Finally, an obstacle path cost $C_O(i)$ is derived from the set of cells (P) connecting the vehicle position with cell i:

$$C_O(i) = \underset{p \in P}{\operatorname{argmax}}(f_O(p)), \tag{9}$$

$$f_O(p) = \begin{cases} 0 & \text{if } \chi_r^p = 1, \\ \kappa_O & \text{otherwise,} \end{cases} \tag{10}$$

where κ_O is a suitably large constant to avoid obstacles. Individually these costs do not produce desirable behavior, however when correctly fused together, the vehicle maintains course. Therefore, the final objective of the goal planning algorithm is to combine the costs and extract the resulting goal χ^G that is to be passed to the motion planning algorithm.

Then to extract goals, we compile a set Ψ that contains the cells with the lowest n% cost, then find a weighted mean over this set:

$$G = \underset{\psi \in \Psi}{\operatorname{argmin}}\Big(\sum_{i=1:N}[||(x_\psi, y_\psi) - (x_i, y_i)|| \cdot C(\psi)]\Big). \tag{11}$$

4. MOTION PLANNING PROBLEM DEFINITION

To reach the goal points generated by the online exploration algorithm, the robot must be able to plan a collision-free path to it. This can be formulated as a motion planning

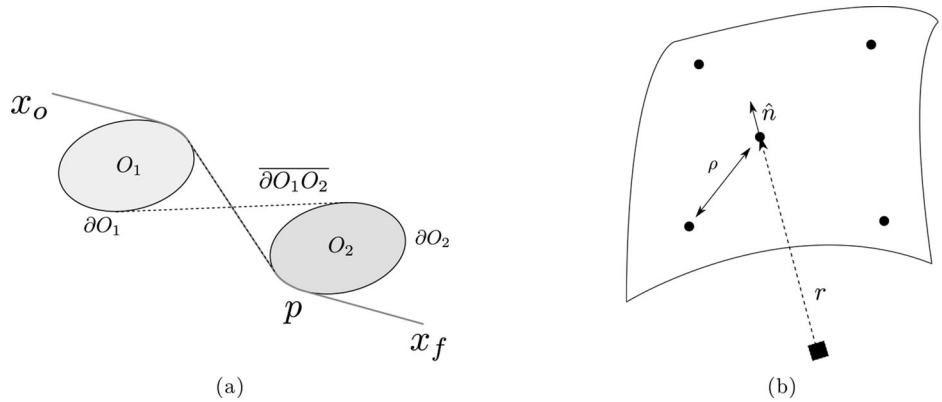


Figure 6. (a) A tangent visibility graph. The optimal solution to the shortest path that does not intersect O_1 and O_2 lies on the graph joining the start x_0 and end x_f to the surface ∂O_1,∂O_2 and the interconnected tangential plane $\overline{\partial O_1 O_2}$. (b) Samples on a tangential surface. The surface is at a distance of r from obstacles. The normal to the surface is $\hat{n}$. Samples on this surface are at least a distance of ρ from each other.

problem to compute the shortest feasible geometric path from start to goal. A velocity profile satisfying deceleration constraints is then assigned to this path to derive a trajectory. This is then given to a trajectory controller similar to that in Hoffmann, Wasl, & Tomlin (2008b). To ensure bounded tracking error, the path is expected to have bounded smoothness.

Let $W \subset \mathbb{R}^3$ be the robot workspace. Let $O \subset W$ be the space where the robot is considered to be in collision. This is the Minkowski sum of the set of obstacles with a sphere of radius of the largest required clearance, r_{obs}. Let a trajectory $\zeta : [0, 1] \to W$ be a well-defined function mapping time to workspace locations. Let the start and goal location be defined as $x_0, x_f \in W \setminus O$. x_0 is the current pose of the robot, while x_f is the goal point generated by the exploration algorithm. If x_f is outside the valid workspace, it is projected back on to $W \setminus O$. Then the planning problem can be formulated as

$$\begin{aligned} \underset{\zeta}{\text{minimize}} \quad & \int_0^1 \|\dot{\zeta}(t)\| \mathrm{dt}, \\ \text{subject to} \quad & \zeta(0) = x_0, \\ & \zeta(1) = x_f, \\ & \zeta(t) \in W \setminus O, \\ & \left|\dot{\zeta}(t)\right| < \varepsilon_{smooth}. \end{aligned} \tag{12}$$

We expand on the nature of W and O for this application. The environment is being updated online by a range-limited sensor. A scrolling grid that is approximately four times larger than the sensor range is maintained. Thus the workspace W is limited to this size. The workspace is an unstructured outdoors environment with varying obstacle density appearing in clusters. The online aspect of the problem requires rapid replanning at a frequency of approximately 10 Hz.

The need for rapid online planning and the clustered distribution of obstacles motivates approaches that sample around obstacles instead of free space. In the next section, we describe an approach that solves the aforementioned problem by constructing a visibility graph around obstacles.

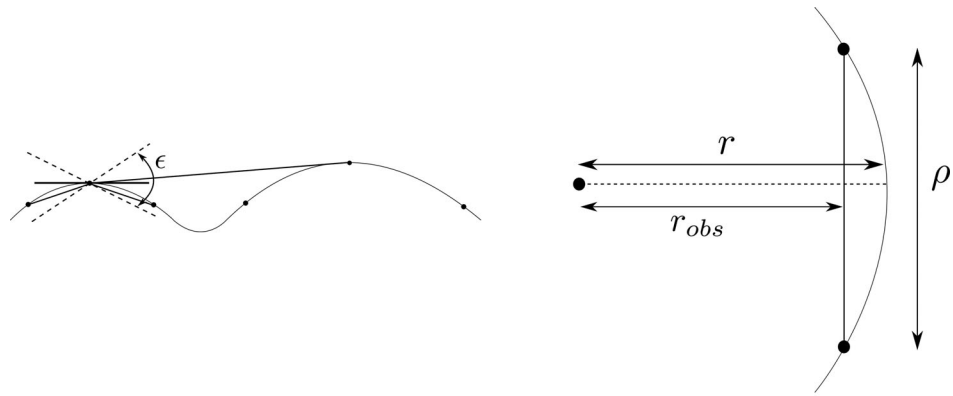
5. MOTION PLANNING ALGORITHM : SPARTAN

SPARTAN (Sparse Tangential Network), which was originally presented in Cover et al. (2013), is a planning approach that creates a sparse graph from vertices that lie on the surface of the collision space O. The optimal path between any two states in the workspace is constrained to lie on ∂O, which is the surface of O, and the tangential plane $\overline{\partial O_i O_j}$, which connects O_i and O_j (proof in Section 6.3) as shown in Figure 6(a). Thus valid edges for the SPARTAN graph are required to satisfy the property of being approximately tangential to ∂O.

SPARTAN is a heuristic driven approximate 3D visibility graph construction technique that solves Eq. (12). Visibility graphs in 3D are NP-hard (Canny, 1988). However, by choosing to accept approximate solutions, SPARTAN samples on the 2D tangential surface and converts the problem to a graph search on a sparse graph. Furthermore, the interleaving of the search and the graph constructions also allows solutions to be computed efficiently.

5.1. Planning Setup

The vertices of the graph constructed by SPARTAN are points sampled on a surface, ideally ∂O. The samples are at least at a distance of ρ from each other, as shown in Figure 6(b). As a result of this discretization, a relaxation of the tangential constraints is required. The edges of the



(a) (b)

Figure 7. (a) A valid edge between two samples on a surface in the SPARTAN graph is tangent to the surface with a tolerance of ϵ. (b) An edge between adjacent samples on the surface of radius r should at least be at a distance of r_{obs} from the center. The samples are at least at a distance of ρ from each other.

SPARTAN graph are required to be approximately tangent to a tolerance of ϵ, as shown in Figure 7(a).

This relaxation violates the constraint $\zeta(t) \in W \setminus O$. To alleviate this problem, the surface to be sampled is inflated to encapsulate ∂O. It is defined as the Minkowski sum of the original obstacle set with a sphere of radius r. Edges approximately ϵ-tangential on this surface do not necessarily violate the obstacle constraints. To constrain the size of the graph, a minimum vertex separation of ρ is ensured. The relation between the resolution ρ, radius of clearance r_{obs}, and the radius of the SPARTAN surface r is such that a connection of adjacent samples does not penetrate O. From Figure 7(b) it can be trivially shown that to satisfy this constraint, the expanded radius $r > \sqrt{r_{obs}^2 + \frac{\rho^2}{4}}$.

The primary optimization objective is to minimize the sum of edge lengths, subject to obstacle constraints. Since the solution path is not smooth, the derivative constraint is translated to a cost function penalizing the angular deviation between consecutive edges. Let the solution be a set of edges $\vec{e}_i$. Then the objective can be expressed as

$$\begin{aligned} \underset{\vec{e}_i}{\text{minimize}} \quad & \sum_{i=0}^{m} \|\vec{e}_i\| + \mu(1 - \vec{e}_{i+1} \cdot \vec{e}_i), \\ \text{subject to} \quad & \vec{e}_i \in W \setminus O. \end{aligned} \tag{13}$$

5.2. Main Algorithm

The notations used in the algorithm are enlisted in Table I. SPARTAN uses A* as the underlying search algorithm. However, since the objective function depends on the dot product between consecutive edges, SPARTAN has to plan in the configuration space. An element of the configuration space $v_c \in \mathbb{R}^3 \times \mathbb{R}^3$ is defined by a workspace vertex and its parent vertex. This creates an additional burden on the size of the priority queue *open* and on the number of possible updates to it. Section 6 presents a method of overcoming this problem, an approach that we have named SPARTAN-lite.

Table I. Notations used in SPARTAN.

Description	Notation
Set of workspace vertices	Σ
Tangential tolerance	ϵ
Priority queue of open configuration space nodes	*open*
Set of closed configuration space nodes	*closed*
Flag indicating expansion of workspace node	*visited*
Set of neighbors to a workspace node	*neighbours*
Parent of a configuration space node	*parent*
Cost of a configuration space node	g
Cost of an edge from a configuration space node	c
Heuristic of a workspace node	h
Start configuration space vertex	v_0
Goal configuration space vertex	v_f

Algorithm 1 explains the SPARTAN process. It follows the same flow as that of A* and only differs in how it efficiently deals with configuration space vertices. It does so by decoupling the cost function into two sections—the part dealing with workspace information and the part dealing with configuration space information. The length and collision checking depend on workspace information. The angle change cost depends on configuration space information.

Lines 14–25 deal with the first visit to a workspace vertex when a configuration space belonging to it is expanded. In this step, tangential and collision checks are performed to determine neighbors to this workspace vertex. Subsequent

visits to this workspace vertex for other configuration space expansions use this information.

Algorithm 1 SPARTAN algorithm

```
procedure SPARTAN(v_0, v_f, Σ)
    g(v_0) = 0
    parent(v_0) = ∅
    open = ∅
    open.insert(v_0, g(v_0) + h(v_0))
    while open ≠ ∅ do
        v_c ← open.pop()
        closed ← closed ∪ v_c
        v_w ← ExtractWorkspace (v_c)
        if v_w = v_f then
            return BuildPath(v_c)
        if ¬visited(v_w) then
            visited(v_w) ← true
            for σ ∈ Σ do
                if IsTangential(v_w, σ, ϵ) then
                    if IsCollisionFree(v_w, σ) then
                        v_new ← (σ, v_w)
                        g(v_new) = g(v_c) + c(v_c, v_new)
                        parent(v_new) = v_c
                        neighbours(v_w) ← neighbours(v_w) ∪ v_new
                        open.insert(v_new, g(v_new) + h(v_new))
    else
        for γ ∈ neighbours(v_w) do
            if γ ∉ closed then
                if g(v_c) + c(v_c, γ) < g(γ) then
                    open.remove(γ)
                    g(γ) = g(v_c) + c(v_c, γ)
                    parent(γ) = v_c
                    open.insert(γ, g(γ) + h(γ))
        return ∅
end procedure
function IsTangential(v_w, σ, ε)
    e⃗ = σ − v_w
    if |GetNormal(v_w).e⃗| < ε and |GetNormal(σ).e⃗| < ε then
        return true
    else
        return false
end function
```

5.3. Implementation Details

The performance of the algorithm is coupled with the ability to sample the vertices on the obstacle surface. The obstacle is represented by a voxel grid that is updated online. The vertex sampling is then done in an efficient way using a modification of incremental distance transform (Cover et al., 2013). The end result is that the surface O is represented in a way that allows efficient collision checking in addition to having vertices on a desired surface. Figure 8(a) shows the distance transform and vertex sampling for a 2D example for ease of visualization. Figure 8(b) shows a 3D planning instance.

5.4. Algorithm Properties

SPARTAN is consistently able to produce high-quality trajectories within a specified computation budget for our use case. In this subsection, we investigate the algorithm properties that explain this performance, and we also shed light on cases that might deter performance.

5.4.1. Completeness

Combinatorial motion planning algorithms are complete because no approximation is made. SPARTAN plans over sampled vertices on the tangential surface, and this approximation prevents it from inheriting the completeness property of the 3D visibility graph. Since SPARTAN imposes an ϵ-tangential criteria on edge connections, resolution completeness is conditioned on ϵ and the SPARTAN planning radius r.

Theorem 5.1. *SPARTAN is resolution complete if* $\epsilon > \cos^{-1} \frac{r_{obs}}{r}$.

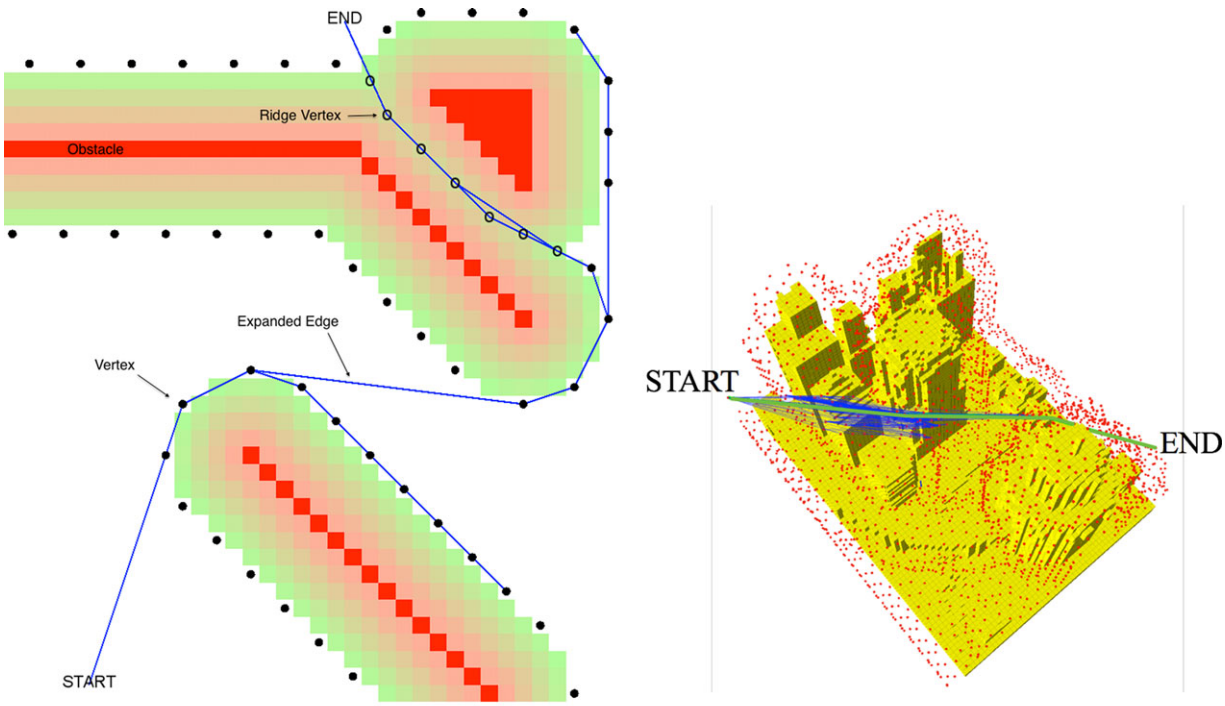
Proof: For resolution completeness, if there exists only one valid sequence of edges $\vec{e}_i$ on Σ joining start and goal, then it must also exist in the SPARTAN graph. In other words, $| n_i \cdot \vec{e}_i | < \epsilon$, where n_i is the normal at the vertices of the path. The limiting case is where e_i activates the feasibility constraint and is tangential to ∂O. To allow this edge, $\epsilon > \cos^{-1} \frac{r_{obs}}{r}$.

5.4.2. Optimality

Since SPARTAN uses an admissible heuristic, to ensure resolution optimality, we are only required to show that an optimal path will always be contained in the SPARTAN graph. The following theorem expands on this.

Theorem 5.2. *SPARTAN is resolution optimal if* $\epsilon > \cos^{-1} \frac{r_{obs}}{r}$.

Proof: This will be proved by negation. Let us assume that the optimal path p is not contained in SPARTAN. Vertices are removed from Σ until p is the only feasible path. SPARTAN is guaranteed to contain p if it is resolution complete. Hence the condition for resolution optimality follows from resolution completeness.



(a) (b)

Figure 8. (a) SPARTAN generates a sparse graph. The obstacles are shown as red (gray scale: dark) squares, the collision space O is colored from red to green (gray scale: dark to light) with decreasing penetration depth, vertices are shown as solid circles, and edges expanded during graph search are shown as lines. (b) SPARTAN creates a sparse graph in three dimensions. The world is represented as a 3D voxel grid. SPARTAN plans on vertices (red dots). The expanded configuration vertices are shown as the connected graph, and the optimal path is shown as a solid green line.

5.4.3. Complexity

Let $\eta(\epsilon) \in [0, 1]$ be the largest fraction of vertices tangential to any given vertex. Let the number of configuration space vertices be $M = \eta(\epsilon)N^2$. Let $|G|$ be the size of the voxel grid used for collision checking. The complexity for the main procedures of SPARTAN is shown in Table II. The total complexity is $O[\eta^2(\epsilon)N^3 \log N]$. The maintenance of configuration space nodes plays a major role.

Table II. Complexity analysis for SPARTAN.

Procedure	Complexity
Addition to Queue	$M \log M$
Tangential Check	N^2
Collision Check	$\sqrt{3}\|G\|M$
Analytic Cost Calculations	$\eta(\epsilon)NM$
Queue Update	$\eta(\epsilon)NM \log M$

6. COMPLEXITY REDUCTIONS IN MOTION PLANNING: SPARTAN-LITE

The major roadblocks in the performance of SPARTAN are due to two main reasons. First, the SPARTAN graph can have dense connectivity when a large tangential surface occurs on which all vertices are interconnected. Second, the maintenance of configuration space vertices leads to a large number of cost function evaluations and queue reordering.

In this section, we will show that both problems can be mediated by exploiting the nature of a geodesic on a smooth manifold. The central idea is that the optimal path on the manifold will not have any deviations—this limits the expansion size of a vertex. This, combined with the fact that the path is constrained to lie on a radius r, implies that optimizing the length is sufficient to ensure that the optimal path has bounded angular deviations. SPARTAN-lite uses these facts to achieve a significant speedup over SPARTAN for difficult planning problem instances.

6.1 Properties of Geodesics and Optimal Path

A *geodesic path* is a locally optimal path and cannot be shortened by slight perturbations. We also define the notion of a plane segment on the tangential surface associated with a vertex as the perpendicular plane to the normal at the vertex and having other plane segments as boundaries. As a result, the plane segments tessellate the tangential surface.

We borrow the definition of planar unfolding from Mitchell, Mount, & Papadimitriou (1987). If the path p connects a sequence of plane segments $\mathcal{P} := \{\mathcal{P}_1, \ldots, \mathcal{P}_k\}$, then the planar unfolding of p along $\mathcal{P}$ rotates the plane segments so as to represent the points of p on $\mathcal{P}_1, \ldots, \mathcal{P}_{k-1}$ in the coordinate frame of $\mathcal{P}_k$.

Theorem 6.1. *If p is a geodesic path which connects the plane segment sequence $\mathcal{P}$, then the planar unfolding of p along $\mathcal{P}$ is a straight line segment.*

Proof: Let $\overline{\alpha x}$, $\overline{x\alpha}'$ be segments on p such that $\angle \alpha x \alpha' \neq \pi$. Then there exists a shortcut $\overline{\beta\beta'}$ for p, with $\beta \in \alpha x$, $\beta' \in x\alpha'$, and the segment $\overline{\beta\beta'}$ lies entirely on the plane sequence $\mathcal{P}$. This contradicts the local optimality property of p. □

Since the edges of the graphs are constrained to pass through the vertices of the plane segment, a relaxation of the straight line criteria is required.

Lemma 6.2. *If p is a geodesic path that connects the plane sequence $\mathcal{P}$ and is constrained to pass through the vertex of the plane sequence, then the angular deviation of a segment on a plane P_i is bounded as $\delta \leq \sin^{-1} \frac{\rho}{l}$, where l is the length of the segment.* □

Proof: Figure 9(a) shows an illustration of the problem. Let α be the projection of e_{i+1} on e_i and Δ be the perpendicular deviation. If $\Delta > \rho$, then there exists v such that $\|v - \alpha\| < \rho$. This removes the need for e_{i+1} to exist. Hence the angular deviation is bounded as $\delta \leq \sin^{-1} \frac{\rho}{l}$, where $l = \|e_{i+1}\|$. □

The optimal path also satisfies properties on the plane on which the normal lies.

Theorem 6.3. *If p is an optimal path that connects the differentiable surface sequence ∂S, then p is always tangential to the surface.*

Proof: If p penetrates the surface ∂S_i, then it violates the planning constraints. If p departs from the surface ∂S_i at x, let $\overline{\alpha x}$, $\overline{x\alpha}'$ be segments on p such that $\overline{x\alpha}' \cdot \partial S_i \neq 0$. Then there exists a shortcut $\overline{\beta\beta'}$ for p, with $\beta \in \alpha x$, $\beta' \in x\alpha'$, and the segment $\overline{\beta\beta'}$ lies in valid free space. This contradicts the local optimality property of p. □

This exact tangential criterion does not occur for approximations of the tangential surface. In addition to the ϵ-tangential criteria, the optimal geodesic can be constrained further.

Lemma 6.4. *If p is an optimal path that connects the plane sequence $\mathcal{P}$ and is constrained to pass through the vertex of the plane sequence, then the outgoing segment angle subtended with the normal at P_i is greater than or equal to the incoming segment angle, subject to relaxation of the obstacle constraint.*

Proof: Figure 9(b) illustrates the problem. Let $\overline{\alpha x} = e_i$, $\overline{x\alpha}' = e_{i+1}$ be segments and $\hat{n}$ be the normal at x such that $e_i \cdot \hat{n} < e_{i+1} \cdot \hat{n}$. Then the segment $\alpha\alpha'$ is a shortcut that does not have a deeper penetration of the plane segment. If $\alpha\alpha'$ intersects with other plane segments, then there exists a feasible path p' approximately tangential to it that is shorter than $\alpha x \alpha'$. □

A further reduction can be applied to constrain the incidence angle.

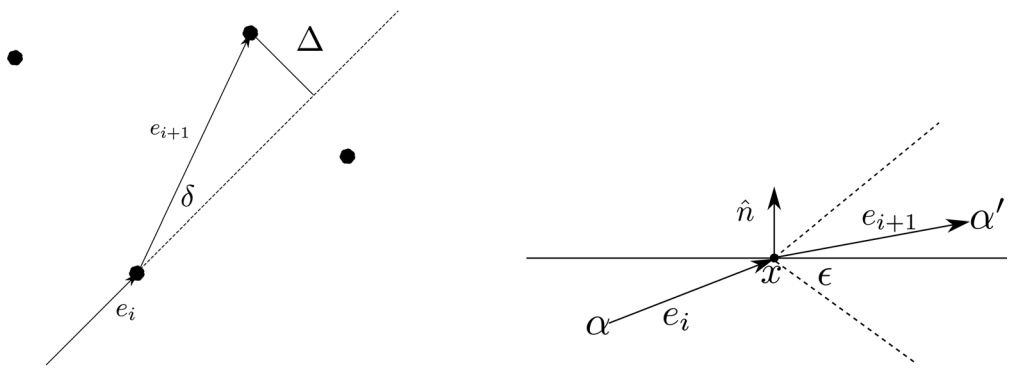
Lemma 6.5. *If p is an optimal path that connects the plane sequence $\mathcal{P}$ and is constrained to pass through the vertex of the plane sequence, then the incoming segment angle subtended with the normal at P_i is acute.*

Proof: Figure 9(b) indicates e_i making an acute angle with the normal. Let $\overline{\alpha x} = e_i$, $\overline{x\alpha}' = e_{i+1}$ be segments such that $e_i \cdot \hat{n} = 0$. If α' is an adjacent vertex to x, then $\|\alpha' - x\| < 2\rho$. Since the graph will not contain αx as an edge (angle not acute), the feasibility of $\alpha\alpha'$ should be examined. As $\|\alpha - x\| \to \infty$, the penetration depth is bounded by $r \cos(2 \sin^{-1} \frac{\rho}{r})$. □

6.2. SPARTAN-lite Algorithm

The SPARTAN-lite algorithm adapts SPARTAN to plan in the workspace and use the geodesic properties in the expansion step. The property of the optimal path to have bounded angular deviation removes the need for a smoothness cost in Eq. (13). This allows planning in the workspace ($\mathbb{R}^3$) in place of the configuration space ($\mathbb{R}^3 \times \mathbb{R}^3$). The SPARTAN expansion step only checked for tangential connections. By applying a constraint on the tangent plane, SPARTAN-lite reduces the connectivity of the graph.

Algorithm 2 shows the main procedure of SPARTAN-lite. It is identical to an A* where the expansion step is defined in line 13.



(a) (b)

Figure 9. (a) On the surface of the tangential plane, angular deviation δ is bounded if the edge sequences e_i and e_{i+1} are part of the optimal path. Δ is the perpendicular deviation of e_{i+1} from e_i. (b) An incident ray on the tangential plane bends away from the normal $\hat{n}$ for a geodesic. e_i is the incoming edge and e_{i+1} is the outgoing edge. ϵ is the tangential tolerance.

Algorithm 2 SPARTAN-lite algorithm

```
procedure SPARTAN-lite v_0, v_f, Σ
    g(v_0) = 0
    parent(v_0) = ∅
    open = ∅
    open.insert(v_0, g(v_0) + h(v_0))
    while open ≠ ∅ do
        v_w ← open.pop()
        closed ← closed ∪ v_w
        if v_w = v_f
            return BUILDPATH(v_w)
        for σ ∈ Σ do
            if ISGEODESIC(v_w, σ, ρ) then
                if σ ∉ closed then
                    if σ ∉ open then
                        g(σ) = ∞
                        parent(σ) = ∅
                    if g(v_w) + c(v_w, σ) < g(σ) then
                        g(σ) = g(v_w) + c(v_w, σ)
                        parent(σ) = v_c
                        if σ ∈ open then
                            open.remove(σ)
                        open.insert(σ, g(σ) + h(σ))
    return ∅
end procedure
```

6.3. Complexity

The complexity analysis is captured in Table III. The total complexity cost for SPARTAN-lite is $O[\xi(\rho)N^2 \log N]$, as compared with the original SPARTAN complexity $O[\eta^2(\epsilon)N^3 \log N]$.

Table III. Complexity analysis for SPARTAN-lite.

Procedure	Complexity
Addition to Queue	$N \log N$
Tangential Check	N^2
Collision Check	$\sqrt{3}\lvert G\rvert\xi(\rho)N^2$
Analytic Cost Calculations	$\xi(\rho)N^2$
Queue Update	$\xi(\rho)N^2 \log N$

7. EXPERIMENTS

We validate our method in a set of experiments, beginning with controlled simulations executed within maps of artificially engineered examples and real-world data. We continue with results from autonomous flights over rivers and waterways, and we also present an open-loop comparison with a human operator based on real-world data.

7.1. Motion Planning Simulation Experiments

The objective of the first set of experiments was to compare SPARTAN to a state-of-the-art motion planner on the planning problems that are likely to occur for river exploration. Given the requirement of producing a smooth 3D path, a generic RRT* was chosen for comparison. A recorded laser scan of an environment was revealed to both planners. The planners were required to plan to a goal point with an average distance of roughly 30 m from the vehicle. Different goal points were used as the vehicle moved through the environment. These goal points were chosen manually with the aim of creating interesting planning problems. Cases with trivial solutions were ignored. Figure 10(a) shows an example of a planning scenario passing through a bridge. The RRT* path was observed to have a low quality in the same planning time as SPARTAN. When running until

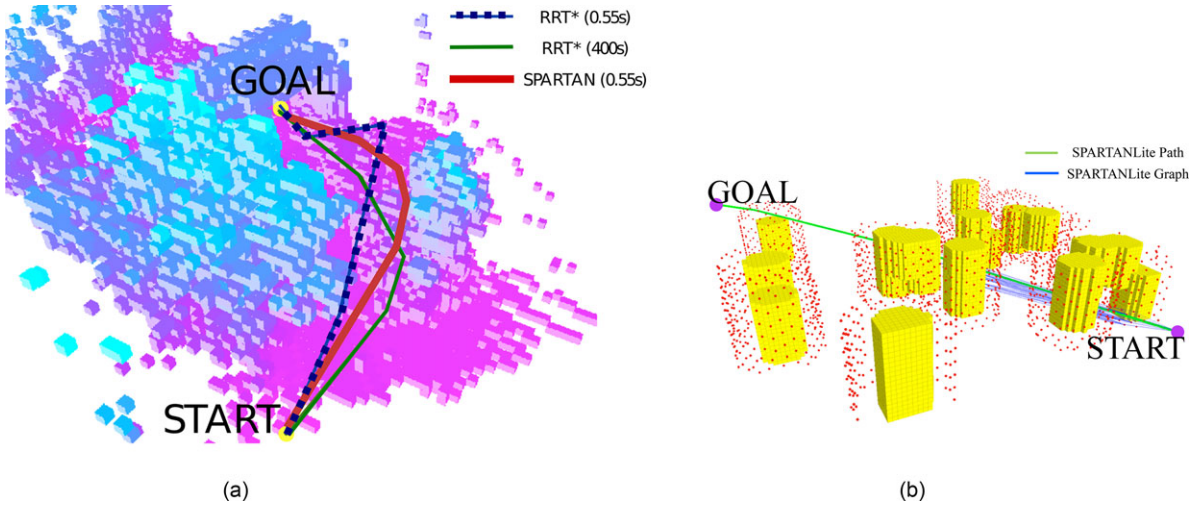


Figure 10. (a) SPARTAN computes a higher-quality path in a smaller time period than RRT*. The environment is an occupancy map from a scanned bridge colored by height above the ground [pink cells are low z and light blue cells are high z (gray scale: dark is low z, light is high z)]. The SPARTAN solution is shown in solid thick red—it has an acceptable path quality and is computed within 0.55 s. SPARTAN-lite computes the same path in 0.38 s. The RRT* solution after 0.55 s is shown in thick dotted blue—the path is of poor quality with large angular deviations. The RRT* path after 400 s is shown in solid thin green—the quality of the path is not significantly different from that of SPARTAN. (b) SPARTAN-lite maintains a sparse graph even in a complicated environment. Then the environment is a Poisson forest of cylinder obstacles whose surface is densely sampled.

Table IV. Comparison for 116 planning problems chosen in an environment constructed from laser scans of an outdoor environment.

Algorithm	Success	Average Cost	Average Time (s)
SPARTAN	100%	2878 (±950)	0.071 (±0.087)
RRT*	76.72%	3472 (±1240)	0.102 (±0.003)

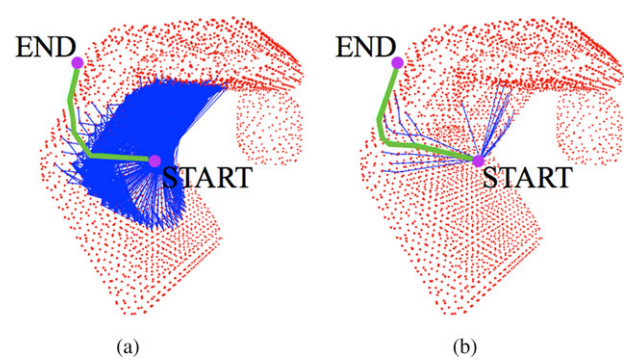


Figure 11. A 3D environment trap where the planners have to plan around a large concave surface. (a) SPARTAN creates a dense graph as many edges are tangential. The planning time is 24.97 s. (b) SPARTAN-lite has a sparse graph due to geodesic constraints. The planning time is 0.476 s.

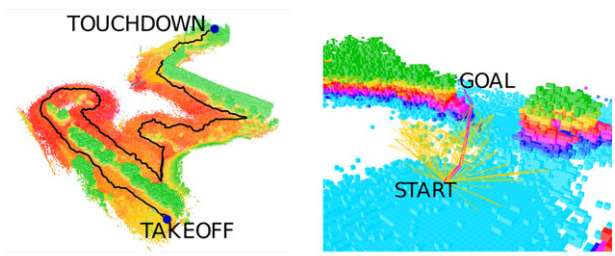
convergence, the path quality did not improve significantly from SPARTAN's output, demonstrating the near optimality of SPARTAN's plan. To analyze the impact of the result in a realistic scenario, both planners are judged on their path quality after 0.1 s (planning cycle of 10 Hz). Table IV shows that SPARTAN always found a solution in the allowed time budget of 0.1 s, while RRT* had a lower success rate. Moreover, SPARTAN's paths have a lower average cost and standard deviation.

The second set of experiments were performed to examine the scenarios in which SPARTAN had a poor performance. Figure 11(a) shows an example in which SPARTAN has an order-of-magnitude slowdown. This is because of the large tangential surface of a concave environment where none of the tangential edges make any progress toward the goal. SPARTAN-lite, in such situations, uses the geodesic properties to create a very sparse graph and is able to find the optimal path approximately 50 times faster.

A benchmark test is run to compare SPARTAN and SPARTAN-lite in a randomly generated environment as shown in Figure 10(b). The environment has a density of $\lambda = 0.00004$, the obstacles are cylinders of radius 5 m and height 20 m, and the goal is chosen at a distance of 141.7 m away. Table V summarizes the results from the test. SPARTAN-lite is on average 2.5 times faster while having a suboptimality bound of 0.02% over SPARTAN.

Table V. Comparison of 100 trials of planning in a Poisson random forest of cylinder obstacles.

Algorithm	Planning Time (s)	Speedup	Path Cost	Suboptimality Bound
SPARTAN	4.066 (±4.642)		141.968 (±2.921)	
SPARTAN-lite	1.716 (±2.074)	2.503 (±1.179)	142.004 (±2.949)	2e-4 (±4.6e-4)



(a) (b)

Figure 12. (a) An autonomous outdoor flight stress-testing the motion planner. The environment is an unstructured one with trees and buildings. The robot was given random goals to emulate an exploration algorithm. The robot explored an area of 1.7 km from takeoff to touchdown. SPARTAN-lite was able to produce collision-free paths while the system flew at an average speed of 1.4 m/s. (b) A particular planning instance during the autonomous stress test run. SPARTAN-lite plans through a clearing in the trees. The search graph is relatively sparse compared to the number of potential edges.

Table VI. Autonomous flight with online motion planning using SPARTAN-lite in a real unstructured outdoor environment.

Planning Time (s)	Vertex Computation (s)	Total Distance (m)	Speed (m/s)
0.175 (±0.199)	0.785 (±0.661)	1744.8	1.395 (±0.431)

7.2. Motion Planning Autonomous Flight Stress Test Experiments

In preparation for full closed-loop autonomy tests, the motion planner was first stress tested for robustness in a real outdoor unstructured environment. The planner was configured to run online on the robot. Figure 12(a) shows the entire experimental run. To emulate the exploration goal planner, pseudorandom goals were provided to the robot as it was flying. The goals were changed every 5 s. The mission speed was on average 1.395 m/s. The entire run spanned 1.7448 km, where the planner was able to always have a collision-free path within a specified planning time. Table VI summarizes the results from the run.

7.3. Exploration Algorithm Simulation Experiments

We evaluate our exploration algorithm in simulation using an environment model generated from data collected over a section of the McCarthy River in Mississippi. A 3D point cloud registered in world coordinates is generated from data collected through the sensor suite carried on a boat traversing the river. From this point cloud, we can simulate the laser measurements given a particular pose of the robot and the known characteristics of our laser scanner. This gives us the means to evaluate our autonomous exploration algorithm based on multivariate cost maps against a traditional frontier exploration algorithm.

We use planning cycles executed every 10 s. Each algorithm is given the same initial conditions and we measure the information gained at each time step during the simulated mission. We repeat the simulation adjusting the virtual sensing range. In Figure 13, the information gained with each approach is plotted against time, where clearly the more traditional frontier approach has difficulties maintaining advantageous trajectories for mapping the riverbank. When narrow passages appear in the river, the frontier algorithm oscillates between returning to explore the earlier unexplored frontier segments and returning to the narrow passages. Our method maintains an optimal distance from the riverbank to avoid suboptimal trajectories, and it also selects trajectories with a higher probability of maintaining course along the river and avoid backtracking down the river to observe portions of the bank.

7.4. Exploration Algorithm Open-loop Comparison

In this section, we compare the exploration algorithm using the vision and laser sensors against a human operator, whom we consider to make expert decisions on how to navigate along the river. Human decisions are either *left* or *right* turn decisions that are measured from heading changes in pose estimate. We perform an open-loop comparison of the sensors and human operator on data collected by the robot platform fixed on a boat driven along the McCarthy River.

We compare the decisions made by the human operator against the turn decisions made based on goal points received from the multi variate exploration algorithm using laser as well as vision sensing.

Since the laser scanner has a short range, laser-based navigation follows the contours of a bank closely making more reactive decisions. The vehicle is able to look further ahead using vision to make intelligent and time-efficient

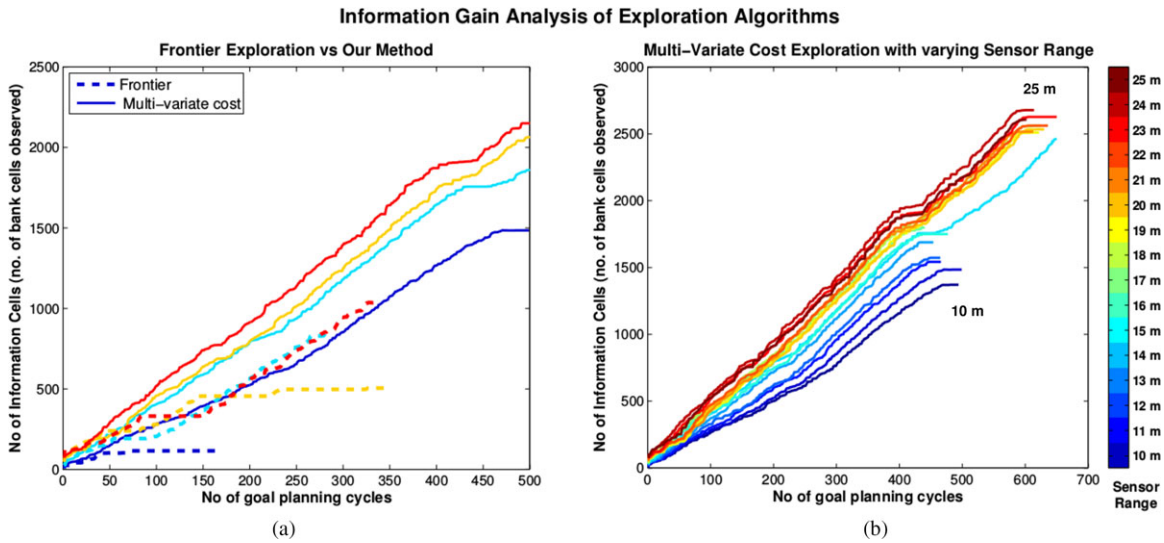


Figure 13. Simulation of autonomous exploration comparing our method using multivariate cost maps against a more traditional frontier exploration algorithm (Yamauchi, 1997). The simulation is given the same initial conditions but varying sensor ranges within an environment model formed from real data collected from a section of the McCarthy River in Mississippi. Left: The information gained using each algorithm is compared using four different sensor ranges. Right: Our algorithm demonstrates more consistent behavior over the different sensor ranges as compared to the frontier exploration algorithm, which oscillates between different unexplored segments of the river. The exploration behavior uniformly degrades with shorter sensor horizons.

decisions. This difference is emphasized in a wide river, where vision will lead the vehicle down the river following its general course, but laser navigation will stick to the bank and follow its contours, which is time-inefficient and in contradiction to what a human operator would do. After an initial 50 m, the next 150 m stretch is more than 25 m wide, and vision performs much better than laser for this stretch; see Figures 14(b) and 14(c).

7.5. Autonomous Flights

After demonstrating in simulation that our motion planning and exploration algorithms are more optimal at returning favorable trajectories, we now proceed to evaluate our approach in real-world truly autonomous flights over rivers and waterways. We manually bring our system into a hover over a river and switch into autonomous mode, and from there we let the vehicle explore the river autonomously at a velocity of 1 m/s with no further human input.

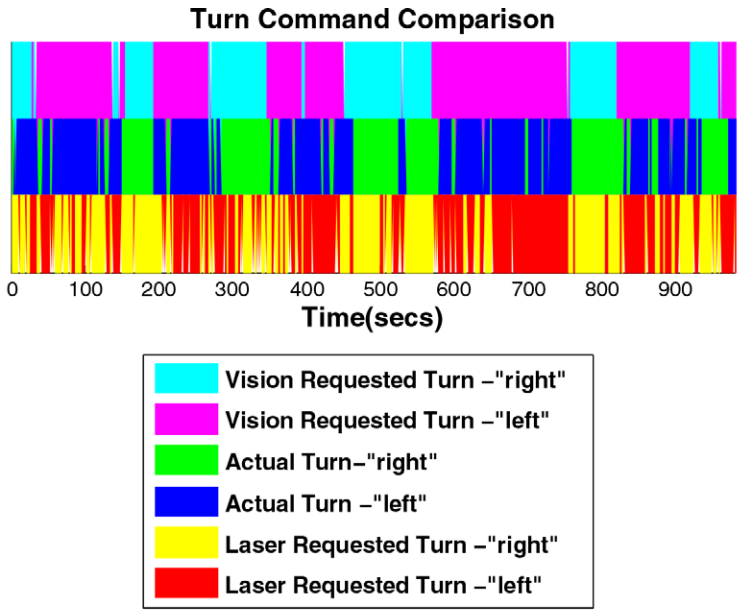
We conduct these autonomous tests in tight and densely vegetated sections of rivers (see Figures 15, 16, and 18) and also on semi-open-water flights conducted along the bank of a lake (see Figure 17).

The flights on tight and densely vegetated sections of a river demonstrate the complete system for planning trajectories that avoid overhanging tree obstacles and maintaining course along the river. The algorithm is able to plan trajectories that enable the system to stay in range of both riverbanks where possible, resulting in an optimal trajectory for maximizing information.

Figure 15 shows an example of the cluttered river environment through which the robot is flying during an autonomous 100 m flight. The canopy and the river/bank classification map from this flight are shown in Figure 16. This experiment demonstrates the advantage of our algorithm against getting information from satellite images, as we are able to classify the areas lying underneath the canopy as river or bank.

We demonstrate the ability of the system to navigate a river over long distances. In the longest autonomous run on a river, the robot flew for about 450 m along the length of a narrow river. The limiting factor in the distance covered during this test was the battery life of the vehicle.

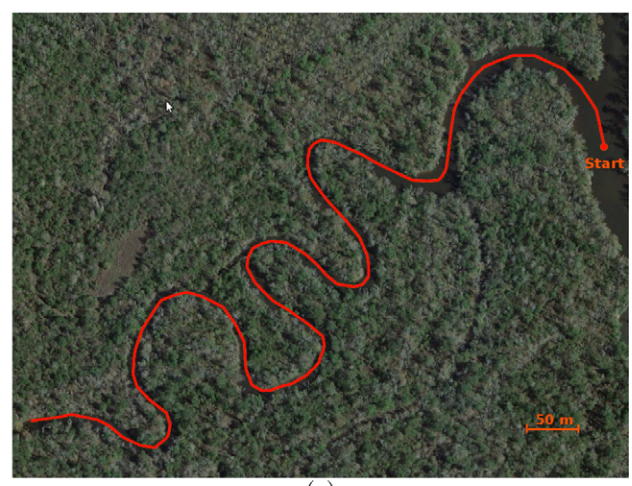
This experiment was conducted on a very shallow river about 10–15 m in width with dense vegetation on both banks. The robot localized itself without any GPS input and was able to classify the cluttered environment into river and obstacles to explore and plan through it. There were some pauses in robot trajectory in some sections of the river due to trees with branches hanging over the river blocking the path for the robot. Shallow areas in the river made the problem harder as they would be classified as obstacles due to the large number of laser returns. Wind was also a challenge as a slight drift from the trajectory would take the robot too close to an obstacle. The final map of the bank and the trajectory of the vehicle during this experiment is displayed in Figure 18. The system operates without manual



(a)

	50-200m	full path
Vision	79.3%	77%
Laser	54%	66%

(b)



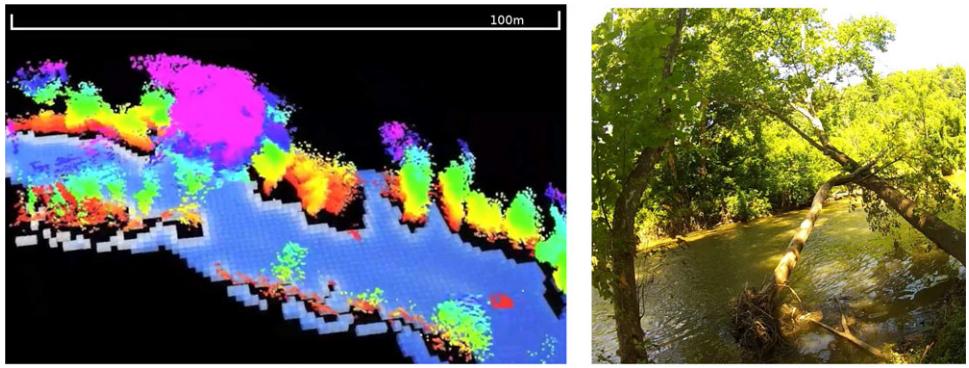
(c)

Figure 14. (a) Automatically extracted turn commands compared against a human operator. Vision commands are more stable and do not require the vehicle to change directions unnecessarily. The laser scanner is much shorter-ranged and the commands are more jittery as they consider a much smaller environment while planning, and they react suddenly to any changes in the shape of the shoreline. (b) % accuracy of vision and laser turn requests by comparing them to the ground truth. Results from the 50–200 m wide stretch show a larger difference in vision and laser accuracies. (c) The 1.5 km path followed on the McCarthy river.

intervention successfully exploring the river and turning according to the river direction.

Longer autonomous flights were conducted on a 600 m stretch of a lake, see Figure 17, where the robot was left to explore the lake bank without any intervention. Frontier exploration algorithms, without any modifications, will lead the vehicle off into the center of the lake, continuously moving toward the largest (or nearest) section of unexplored frontier. Our approach using a variety of costs will enable the vehicle to seek to maximize the length of the bank explored.

Autonomous flights in both tight and open water situations highlight the ability of our algorithm to maintain desirable behaviors in a variety of conditions. Dense overhanging trees demonstrate the fast motion 3D planning algorithm ability to rapidly replan paths around 3D obstacles.



(a) (b)

Figure 15. (a) Laser point cloud collected from the autonomous flight through the densely forested river environment. The detected river extent is shown in grid cells, and the bank and overhanging trees are colored by height above the river surface. (b) Environment where we test the system's ability to fly autonomously through a densely forested river.

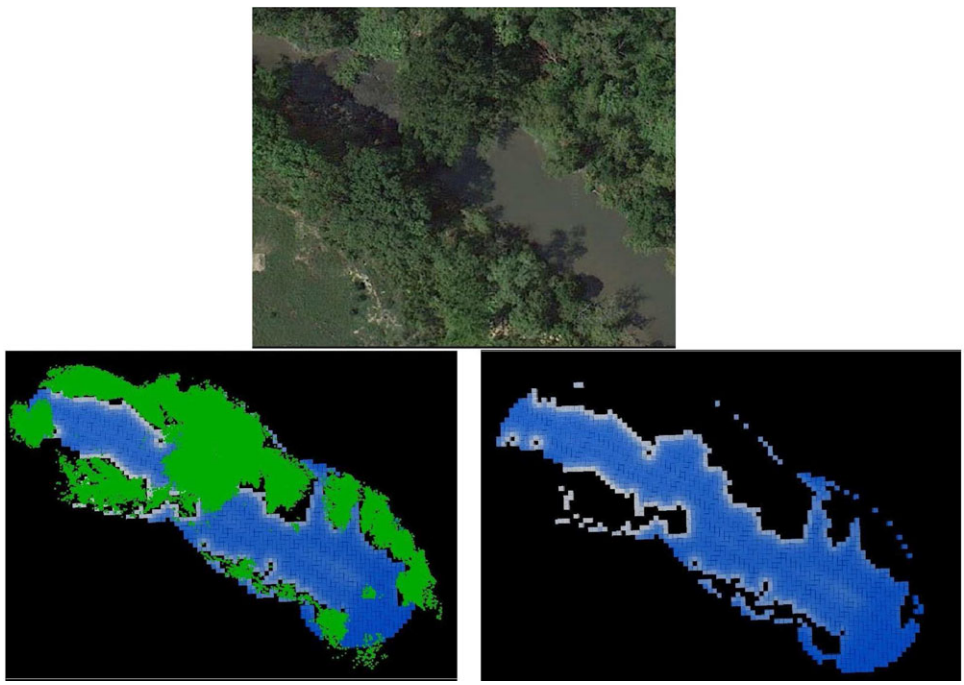


Figure 16. Map generated from autonomous flight through a narrow and densely vegetated river-segment. Top: satellite view of the river segment. Left: map of the river, where the river extent is shown in gridded cells. Overhanging trees and bank detected by the laser scanner are shown as a point cloud. Right: traversable river extent detected by the laser scanner. Notice that the overhanging trees in the middle of the segment are removed, and a clear and traversable path is discovered underneath by the flying vehicle. The traversable river map, for example, could be used by a surface craft following the flying vehicle, enabling it to have knowledge of where it is safe to travel.

The exploration algorithm was seen to stay an optimal distance from the bank to maintain good coverage with the laser scan pattern, and it did not wonder away exploring the large internal area of the lake.

7.6. Video

A video of the longest flight (450 m) on a tight passage of a bending river can be seen at https://www.youtube.com/watch?v=vaKNbzYSK6U.

Figure 17. Satellite image with overlaid flight trajectory (white) and bank map (green point cloud) collected from a 660 m autonomous flight over water along the bank of a lake.

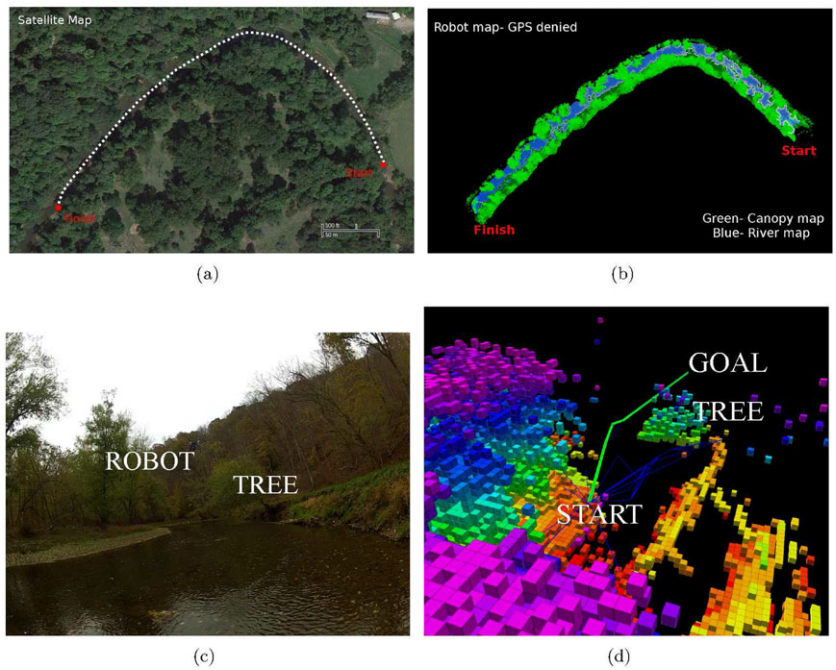


Figure 18. Autonomous flight: (a) Satellite image with overlaid flight trajectory and (b) the river/bank map (blue) and canopy map (green) generated by the robot from data collected in a 450 m autonomous flight along a river. (c) SPARTAN-lite planning around a tree. The exploration algorithm commands a goal at a point beyond the tree. The robot has to plan a safe trajectory between the canopy on either side of the bank. (d) The occupancy grid view of the scenario in (c). Due to the resolution of the obstacle map, SPARTAN-lite has to plan through a narrow space. A large tangential space is present, however SPARTAN-lite creates edges in its graph that can only be a part of the optimal solution. This results in the creation of a very sparse graph, and the planner is able to compute a solution quickly.

8. LESSONS LEARNED

A three-year program with adventurous goals taught our team many lessons.

Early in the program, we realized frequent field testing in candidate environments was essential to discover the core problems quickly, iterate on the weaknesses of the system, and arrive at a successful demonstration of a GPS-denied river-exploring robot. Our field work began with data collection with one or two key sensors to initiate development of the core perception system. Once initial perception design was finalized, the sensor hardware was iterated to a full sensor prototype. Then full data collection was conducted with realistic traverses of riverways, after which the sensor payload was miniaturized and mounted onto the micro air vehicle frame such that data collection could be done with manually piloted flights. After manual flights, we progressed to autonomous flights over land. Finally, once the system was reliable over land, we progressed to autonomous flights over water.

We discovered through the program that high performance could be achieved by taking advantage of the semantics of the environment where necessary. For example, the laser response characteristics could be used with the river surface to estimate the height above the river using specular

returns, and to detect the presence of the river extent. In addition, we exploited the horizon line for the self-supervised segmentation algorithm by identifying known nonriver regions of the image and using the slightly different color and texture of the river in the camera image.

The exploration algorithm began with some simple requirements that were intended to maximize the length of the river bank that was explored with a fixed, battery-limited mission duration. This led to insights into what pieces of information should be considered by the algorithm. In our case, we discovered that the algorithm should use time, distance from vehicle, and distance from obstacles simultaneously.

We have found that online 3D trajectories could be computed in real time by exploiting smoothness in the obstacle distance transform to drastically reduced search space.

In terms of hardware, the larger frame vehicle of the Oktokoptor enabled a generous 1 kg payload, which made it possible to carry larger processing hardware that substantially simplified real-time algorithm development. The larger payload also enabled the miniature spinning 3D spinning laser scanner to be carried on a micro air vehicle, which may well be the first example of a 3D laser scanner carried on such a vehicle.

We have found that even with drifting odometry that had substantial latency (about 200 ms), the combination of using an effective UKF filter and a somewhat local and reactive exploration approach produced a compelling and successful autonomous demonstration in a challenging and risky environment.

9. CONCLUSION

Our work demonstrates that autonomous exploration is possible in river environments and that neither GPS waypoints nor prior maps are necessary. Such an autonomous system necessitated that all parts of the system meet all requirements for weight, efficiency, and robustness. In several of the system components, this required state-of-the-art development, including the miniature 3D laser scanner, the river detection algorithms, the positioning system and planning, and exploration algorithms. A key outcome, and the focus of this paper, was the development and demonstration of the two algorithms to robustly extract goal points, and efficiently planning 3D trajectories in challenging unstructured terrain. Results indicate that the two algorithms surpass the performance of other commonly deployed techniques. While there is much work in autonomous exploration for ground-vehicles, these algorithms do not directly translate to river environments. Our system is developed to respect the specific physical layout and properties of the river and bank and the behavior of the sensors and perception algorithms in these environments.

In future work, we still see challenges to increase the operating velocity in a safe manner. One avenue to explore is to predict the course of the river to forecast which directions the river will follow when turns, dead-ends, or forks are likely to appear. We also see persistent monitoring of a waterway as an important means to detect pertinent changes to the environment. Further, we see that, when combined with a supporting surface vehicle that is larger and trails behind, the flying vehicle has the ability to return for landing and recharging. These nonhomogeneous teams of vehicles pose many interesting research challenges, such as high-fidelity localization and tracking and with relative motion planning for high-speed takeoffs and landings, in addition to information sharing to exploit the different sensing characteristics and viewing perspectives of the vehicles.

ACKNOWLEDGMENTS

The work described in this paper is funded by the Office of Naval Research under Grant No. N00014-10-1-0715.

REFERENCES

Achar, S., Sankaran, B., Nuske, S., Scherer, S., & Singh, S. (2011). Self-supervised segmentation of river scenes. In 2011 IEEE International Conference on Robotics and Automation (pp. 6227–6232). IEEE.

Amigoni, F., & Caglioti, V. (2010). An information-based exploration strategy for environment mapping with mobile robots. Robotics and Autonomous Systems, 58(5), 684–699.

Canny, J. (1988). Some algebraic and geometric computations in PSPACE. In Proceedings of the 20th Annual ACM Symposium on Theory of Computing (pp. 460–467). ACM.

Chambers, A., Achar, S., Nuske, S., Rehder, J., Kitt, B., Chamberlain, L., Haines, J., Scherer, S., & Singh, S. (2011). Perception for a river mapping robot. In 2011 IEEE/RSJ International Conference on Intelligent Robots and Systems (pp. 227–234). IEEE.

Chambers, A. D., Scherer, S., Yoder, L., Jain, S., Nuske, S. T., & Singh, S. (2014). Robust multi-sensor fusion for micro aerial vehicle navigation in gps-degraded/denied environments. In Proceedings of the American Control Conference, Portland, OR.

Cover, H., Choudhury, S., Scherer, S., & Singh, S. (2013). Sparse tangential network (spartan): Motion planning for micro aerial vehicles. In International Conference on Robotics and Automation.

Dunbabin, M., Grinham, A., & Udy, J. (2009). An autonomous surface vehicle for water quality monitoring. In Australasian Conference on Robotics and Automation (ACRA).

Ferguson, D., Likhachev, M., & Stentz, A. (2005). A guide to heuristic-based path planning. In Proceedings of the International Workshop on Planning under Uncertainty for Autonomous Systems, International Conference on Automated Planning and Scheduling (ICAPS).

Gadre, A., Du, S., & Stilwell, D. (2012). A topological map based approach to long range operation of an unmanned surface vehicle. In American Control Conference, June (pp. 5401–5407).

Geiger, A., Ziegler, J., & Stiller, C. (2011). Stereoscan: Dense 3d reconstruction in real-time. In Intelligent Vehicles Symposium (IV), 2011 IEEE. 5–9 June (pp. 963–968).

Goerzen, C., Kong, Z., & Mettler, B. (2010). A survey of motion planning algorithms from the perspective of autonomous uav guidance. Journal of Intelligent & Robotic Systems, 57(1), 65–100.

Hoffmann, G., Waslander, S., & Tomlin, C. (2008a). Quadrotor helicopter trajectory tracking control. In AIAA Guidance, Navigation and Control Conference and Exhibit, Honolulu, HI.

Hoffmann, G. M., Wasl, S. L., & Tomlin, C. J. (2008b). Quadrotor helicopter trajectory tracking control. In Proceedings of the AIAA Guidance, Navigation, and Control Conference.

Hrabar, S. (2011). Reactive obstacle avoidance for rotorcraft uavs. In 2011 IEEE/RSJ International Conference on Intelligent Robots and Systems (pp. 4967–4974). IEEE.

Jain, S., Nuske, S., Chambers, A., Yoder, L., Cover, H., Chamberlain, L., Scherer, S., & Singh, S. (2013). Autonomous river exploration. In Proceedings of the International Conference on Field and Service Robotics. Springer Tracts in Advanced Robotics 105, Springer 2015, ISBN 978-3-319-07487-0.

Karaman, S., & Frazzoli, E. (2010). Incremental sampling-based algorithms for optimal motion planning. In Proceedings of Robotics: Science and Systems Conference online proceedings. http://arxiv.org/abs/1005.0416.

Kavraki, L., Svestka, P., Latombe, J., & Overmars, M. (1996). Probabilistic roadmaps for path planning in high-dimensional configuration spaces. IEEE Transactions on Robotics and Automation, 12(4), 566–580.

Kendoul, F. (2012). Survey of advances in guidance, navigation, and control of unmanned rotorcraft systems. Journal of Field Robotics, 29(2), 315–378.

Kollar, T., & Roy, N. (2008). Trajectory optimization using reinforcement learning for map exploration. The International Journal of Robotics Research, 27(2), 175–196.

Kuwata, Y., & How, J. (2004). Three dimensional receding horizon control for uavs. In AIAA Guidance, Navigation, and Control Conference and Exhibit (Vol. 3, pp. 2100–2113).

LaValle, S. (1998). Rapidly-exploring random trees: A new tool for path planning. TR 98-11, Computer Science Dept., Iowa State University.

LaValle, S. (2006). Planning algorithms. Cambridge: Cambridge University Press.

Leedekerken, J., Fallon, M., & Leonard, J. (2010). Mapping complex marine environments with autonomous surface craft. In The 12th International Symposium on Experimental Robotics. Springer Berlin Heidelberg, 2014.

Mitchell, J. S., Mount, D. M., & Papadimitriou, C. H. (1987). The discrete geodesic problem. SIAM Journal on Computing, 16(4), 647–668.

Moorehead, S. J., Simmons, R., & Whittaker, W. L. (2001). Autonomous exploration using multiple sources of information. In IEEE International Conference on Robotics and Automation.

Pradalier, C., Posch, T., Pernthaler, J., & Siegwart, R. (2012). Design and application of a surface vessel for autonomous inland water monitoring. IEEE Robotics & Automation Magazine (pp. 1–9).

Rathinam, S., Almeida, P., Kim, Z., Jackson, S., Tinka, A., Grossman, W., & Sengupta, R. (2007). Autonomous searching and tracking of a river using an UAV. In American Control Conference (pp. 359–364). IEEE.

Rehder, J., Gupta, K., Nuske, S., & Singh, S. (2012). Global pose estimation with limited gps and long range visual odometry. In Proceedings of the 2012 IEEE/RSJ International Conference on Robotics and Automation.

Scherer, S., Rehder, J., Achar, S., Cover, H., Chambers, A., Nuske, S., & Singh, S. (2012). River mapping from a flying robot: State estimation, river detection, and obstacle mapping. Autonomous Robots, 33(1-2), 189–214.

Scherer, S., Singh, S., Chamberlain, L. J., & Elgersma, M. (2008). Flying fast and low among obstacles: Methodology and experiments. The International Journal of Robotics Research, 27(5), 549–574.

Stachniss, C., & Burgard, W. (2003). Exploring unknown environments with mobile robots using coverage maps. In International Joint Conference on Artificial Intelligence (pp. 1127–1132).

Tsenkov, P., Howlett, J., Whalley, M., Schulein, G., Takahashi, M., Rhinehart, M., & Mettler, B. (2008). A system for 3d autonomous rotorcraft navigation in urban environments. In AIAA Guidance, Navigation and Control Conference and Exhibit, Honolulu, HI.

Valada, A., Velagapudi, P., Kannan, B., Tomaszewski, C., Kantor, G. A., & Scerri, P. (2012). Development of a low cost multi-robot autonomous marine surface platform. In The 8th International Conference on Field and Service Robotics. Springer Berlin Heidelberg, 2014.

Whalley, M., Tsenkov, P., Takahashi, M., Schulein, G., & Goerzen, G. (2009). Field-testing of a helicopter UAV obstacle field navigation and landing system. In 65th Annual Forum of the American Helicopter Society, Grapevine, TX.

Wooden, D., & Egerstedt, M. (2006). Oriented visibility graphs: Low-complexity planning in real-time environments. In Proceedings of the 2006 IEEE International Conference on Robotics and Automation (pp. 2354–2359). IEEE.

Yamauchi, B. (1997). A frontier-based approach for autonomous exploration. In 1997 IEEE International Symposium on Computational Intelligence in Robotics and Automation (pp. 146–151). IEEE Computer Society Press.