Autonomous Exploration for Infrastructure Modeling with a Micro Aerial Vehicle

Luke Yoder and Sebastian Scherer

Abstract Micro aerial vehicles (MAVs) are an exciting technology for mobile sensing of infrastructure as they can easily position sensors in to hard to reach positions. Although MAVs equipped with 3D sensing are starting to be used in industry, they currently must be remotely controlled by skilled pilot. In this paper we present an exploration path planning approach for MAVs equipped with 3D range sensors like lidar. The only user input that our approach requires is a 3D bounding box around the structure. Our method incrementally plans a path for a MAV to scan all surfaces of the structure up to a resolution and detects when exploration is finished. We demonstrate our method by modeling a train bridge and show that our method builds 3D models with the efficiency of a skilled pilot.

1 Introduction

The goal of this work is to use a MAV to rapidly model large outdoor structures with arbitrary geometry. As-built 3D models are increasingly used in a number of industries to detect structural problems, assess damage, design renovations, and to organize other types of data. Although the industry standard ground-based lidar are accurate, building models with them is slow and they suffer from occlusion problems. Large structures found in outdoor, open air environments present an excellent application for MAV-based lidar which can reach almost any vantage point. Compared to stationary ground-based lidar, MAV-based lidar output a model with lower resolution and higher uncertainty as the vehicle must use lightweight sensors and must estimate its position. The benefit of a MAV-based lidar is coverage and rapid

Carnegie Mellon University, Pittsburgh, PA, USA
e-mail: lyoder@cmu.edu, basti@cmu.edu

This research is funded by the National Science Foundation under grant IIS-1328930

deployment. In many environments flying scanners can reach the vantage points required to achieve complete coverage, and do so very quickly. Rather than competing with existing terrestrial lidar, we imagine such a system supporting new applications that require low resolution and complete coverage point clouds.

A number of MAV systems are commercially available for building 3D models of outdoor environments. Acquiring data with these systems, however, is still a manual process requiring a skilled pilot. Not only is safely piloting a MAV near a structure difficult, scanning complex structures under manual control is prone to error as it is difficult for a human pilot to remember what has and what has not been scanned. We propose a practical solution where a user draws a 3D bounding box around a structure then a small flying vehicle autonomously scans all surfaces of the structure, producing a 3D point cloud.

One challenge in 3D modeling infrastructure with an autonomous MAV is developing path planning algorithms. If a prior model of the infrastructure is available, "inspection" or "coverage" planning could lead the MAV to efficiently scan all surfaces [7, 2]. In many applications, however, obtaining a prior model is infeasible. Requiring a prior map will limit the adoption of a robotic planning algorithm for infrastructure modeling. Without a priori models to use for path planning, the remaining problem is one of choosing an exploration path through a partially observed environment with the goal of maximizing exploration efficiency.

The contribution of this paper is a simple yet effective 3D path planning algorithm for completely scanning complex 3D environments with a range sensor attached to a MAV. Specifically, we present the surface frontier, a fundamental geometric aspect of 3D surface exploration, and we present an incremental path planning algorithm using surface frontiers to guide the observation of unknown surface until the surface is completely detected. Finally, we present real world results showing that our system performs as well as or better than a skilled pilot.

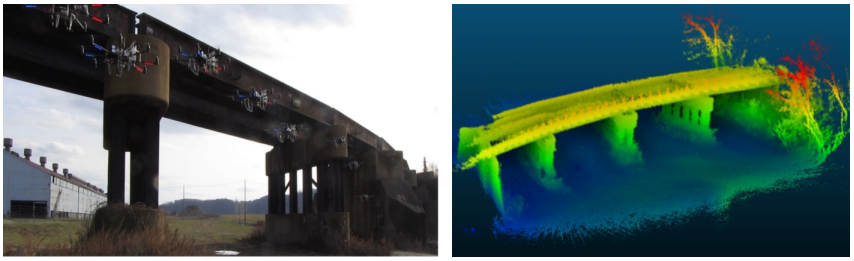

Fig. 1 A MAV autonomously modeling a structure (left) and the resulting point cloud (right)

2 Related Work

Work related to ours does not rely on a prior model of the environment but iteratively plans exploration paths through a partially observed environment. Common to almost all of the following approaches is discretizing the continuous world into an occupancy grid where cells in the occupancy grid are either known occupied, known free, or unknown to the robot. The difference between most of the following methods is in how the information in the occupancy grid is used to guide exploration.

Frontier exploration [16] is a simple 2D exploration algorithm used extensively on ground robots. In frontier exploration, information is gained by traveling to frontiers, a heuristic that has proven successful at guiding a robot to a vantage point where unobserved cells can be observed. Extensions of frontier exploration to 3D without reducing occupancy grid resolution are too computationally costly [5] to run in real time on a computationally constrained MAV. Shade et al. [12] propose a 3D frontier exploration and integrated path planning algorithm that runs in real time, but is designed for exploration of free space not modeling outdoor surfaces. Several groups achieve outdoor exploration and 3D mapping using MAVs by limiting exploration to 2D. Heng et al. [6] implement frontier based exploration and wall following exploration which they validate in outdoor urban environments. Jain et al. [8] propose a frontier shoreline exploration algorithm which they validate on a MAV exploring rivers.

Some methods estimate entropy reduction over a set of possible paths to determine the best exploration path. Stachniss et al. [15] propose generating paths that lead to frontiers and paths that lead to previously visited locations to improve localization. The path that minimizes the sum of map entropy and pose entropy is chosen.

"Next best view" planning algorithms have been developed over the last three decades [9, 11] to incrementally plan sensor views for modeling objects. Next best view algorithms generally constrain the search space of views around the object and then find a view that maximizes some utility function. The utility function for evaluating views might include unknown cells visible from the view and overlap with previously acquired data for data registration.

To reduce computational cost, a number of techniques attempt to find exploration goals without managing an occupancy grid. One method designed for indoor 3D exploration [13] simulates particles expanding from free space according to Newtonian dynamics. Exploration goals are set in regions of high expansion. This algorithm is designed as a simplification of frontier exploration allowing the authors to successfully explore indoor environments with a computationally constrained MAV. Another simulated particle based method [1] simulates liquid falling on a 3D outdoor scene and detects exploration goals by finding areas where the simulated liquid gets through the point cloud. This algorithm is limited to exploring 2.5D terrain because the algorithm is not designed for detecting holes in vertical walls and under overhangs.

Our method is closely related to next best view techniques although it employs a simpler utility function. Our utility function is based on the visibility of 2D frontiers on the 2D surface of a 3D object, which are related to the frontiers used in frontier exploration.

3 Problem Formulation

This section provides a representation for the state of a MAV and a partially observed environment. In this section we also describe the problem that we go on to solve in Section 5, as well as some assumptions.

Since we are exploring outdoors with the ability to navigate in 3D, we start by constraining exploration to a region of interest $R \subseteq W$ where $W \subseteq \mathbb{R}^3$ is the world. The MAV's position in W is defined by its state $X = [x, y, z, \varphi, \theta, \psi, c]$ where (x, y, z) and (φ, θ, ψ) are position and orientation in world coordinates, respectively. The variable c is the configuration of the sensors on the vehicle which can be multidimensional. We will refer to proposed trajectories T which symbolize time parametrized state such that $X_i = T(t_i)$ for $t_i \in [t_o, t_f]$.

We represent W as a 3D occupancy grid where cells C^j can be unknown C_u, free C_f, or occupied C_o. The goal of our exploration planning algorithm is to classify all observable C_o in R. Supporting this goal, we define surface information as

$$I_s = \sum_{j=1}^{m} \begin{cases} 1, & \text{if } C^j = C_o \text{ and } C^j \subseteq R \\ 0, & \text{otherwise} \end{cases} \tag{1}$$

Where m is the number of cells in R. We define I_s^* as the surface information when all surfaces in the environment are observed. The problem that we would like to solve is to find an optimal trajectory T^* from which the sensor observes all surfaces (i.e. $I_s = I_s^*$) in minimum time. Since an optimal coverage path in an environment that is only partially observed may not be possible to compute, we are left with the goal of achieving high exploration efficiency in terms of surface information gain per unit time.

We need to make a few assumptions. First, we assume that free space in R is one connected set reachable without leaving R. Second, we assume the vehicle's state estimation is accurate and robust enough to not require active localization. Some exploration methods [15] estimate the MAV's ability to localize over proposed trajectories to help maintain low uncertainty in the map. We find that localization accuracy around large structures using our laser odometry approach [17] is accurate enough for exploration using large occupancy grid cells. For this work we assume that active localization is not necessary.

We use the occupancy grid representation to describe our method in Section 5 but first we describe the surface frontiers that guide our exploration approach.

4 Surface Frontiers

Our goal is to sense a structure's surface, not the free space around the structure. Traditional frontier exploration would lead a MAV to explore free space in addition to surface. Because of this, frontier exploration in an outdoor environment may be inefficient. Our method, proposed in Section 5, prioritizes surface modeling by using the boundary between known and unknown surface to guide exploration. In this section we describe topological properties of known and unknown space that motivate our method. This section starts by introducing the concept of a surface frontier, then goes on to discuss its benefit for exploration.

4.1 Definition

We consider W as topological space where all points $p \in W$ are either occupied p_o or free p_f. All p are static and all p_o belong to one connected set $P_o = \{p_o\}$. Since W is either unobserved or partially observed a priori, points unknown to the MAV are designated p_u and known points p_k. In this section we assume the MAV is equipped with an ideal range sensor capable of classifying unoccluded volume in its field of view as occupied or free. We define p_k^* as points that can be observed from the free space connected set that the MAV navigates through. As described in Section 3, we assume $p_k^* \cap R$ is one connected set. Since a range sensor can only observe the surface of occupied space,

$$S_k^* = P_o \cap \{p_k^*\}$$

is a surface which does not change during exploration. S_k^* is the object surface observable from the free space connected set that the MAV navigates through, though not necessarily observable by the MAV if some free space is not reachable. During exploration a subset of S_k^* are the known surfaces

$$S_k = P_o \cap \{p_k\}$$

Since S_k is a subset of the surface S_k^* during exploration, we know that the boundary of known surface ∂S_k is also a boundary of unknown surface, where ∂ is the boundary operator. In other words, an unobserved surface must be present just beyond the known surface boundary. We call ∂S_k surface frontiers. During exploration, ∂S_k is a set of one dimensional manifolds as shown in Figure 2. If we can guide a robot to observe surface frontiers, it is almost certain that we will observe unknown surface.

4.2 Surface Frontier Exploration

For practical purposes we would like a simple way to define what volume in the world should be explored. Then, given a volume to be explored, we would like to detect when exploration is complete. This subsection introduces a region of interest bounding the volume to be explored, then shows how we can terminate exploration when all surfaces in the volume are observed without exhaustively exploring unknown space.

We set a region of interest grouping points into the connected set R which is the volume containing the structure to be modeled. As an example, R could be the volume inside a cuboid defined by someone using the system. The goal is to observe all surfaces inside the bounding box $S_k^* \cap R$.

We define the region of interest boundary ∂R intersecting the connected free space $\{p_k^*\} \cap \{p_f\}$ as the observable region of interest boundary

$$B = \partial R \cap \{p_k^*\} \cap \{p_f\}$$

Combining surface B with all object surfaces observable by the MAV gives us the surface to be explored

$$E = B \cup S_k \cap R$$

If $\{p_k \in R\} = \{p_k^* \in R\}$ then E becomes a closed connected surface denoted E^*. We can use this property to determine when exploration is complete. If, during exploration, E is a set of connected surfaces instead of a closed connected surface, exploration is not complete as there still may be unobserved surfaces in the bounding box.

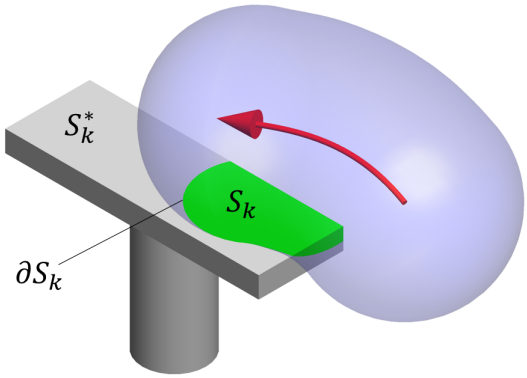

Fig. 2 Surface frontiers at the beginning of exploration

Alternatively, if we assume that $S_k^* \cap R$ is a single connected surface then exploration is complete when $\partial S_k^* \cap R$ is an empty set. This assumption relieves the MAV of having to exhaustively search B.

Only using surface frontiers to guide exploration means we are not making any assumptions about a structure's geometry. Including a priori information about the structure (e.g. max curvature) or other heuristics (e.g. number of C_u visible from a view) could improve performance but would also increase algorithm complexity and possibly reduce generality.

5 Method

In the following we formulate an exploration planning strategy for a MAV tasked with the exploration problem introduced in Section 3. Unlike the frontier exploration algorithm [16], we cannot directly navigate to surface frontiers as doing so might lead to a collision. We also may not be able to observe frontiers if the MAV cannot reach a state where the frontier is visible. To use surface frontiers for exploration, we compute $T(t_i)$ from which we can observe frontiers, and terminate exploration when surface frontiers are not observable by the MAV. In this section we start by describing how to detect surface frontiers in the occupancy grid. We then introduce a utility function for estimating the utility of views for sensing surface frontiers. Finally, we describe how we can plan exploration paths using an objective function that trades off between a view's utility and the path cost of navigating to the view.

5.1 Occupancy Grid Surface Frontiers

Given an occupancy grid with occupied, free, and unknown cells, surface frontiers can be detected by finding connections between all three cell classes. An occupancy grid surface frontier can be defined as a free cell C_f with a known occupied neighbor

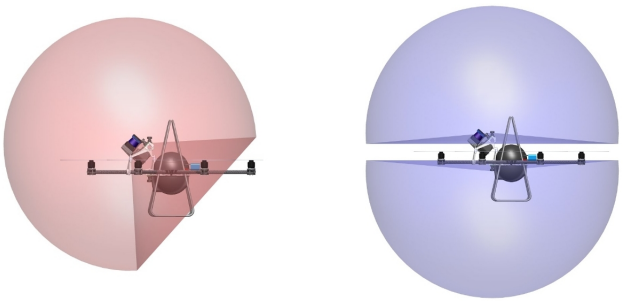

Fig. 3 lidar field of view (left) and camera field of view (right)

C_o and an unknown neighbor C_u where C_o and C_u are also neighbors. An example of surface frontiers in the occupancy grid representation is shown as the blue cells in Figure 6.

5.2 View Utility

To guide the observation of surface frontiers we create an exploration utility function that estimates the number of surface frontier observations that can be made at a given view.

First we make a simplification based on our vehicle's sensor configuration. Our vehicle's lidar is nearly omni-directional as shown in Figure 3. Only a small blind spot is present behind the vehicle. We assume that the lidar scans in a spherical pattern sampling in a circular uniform distribution, which allows us to reduce the degrees of freedom of the sensor field of view. Assuming omni-directional sensing allows us to simplify our MAV's state to $X = [x, y, z]$.

Assuming one complete scan is taken we can approximate the number of lidar rays that will hit one unoccluded surface frontier cell. To use this as a utility function we also consider safe flying distance from the structure d_s and a maximum desired measurements per cell t_m. Given the cell height h, the distance from the sensor to the cell r, and the number of points per scan N, the utility of a view for observing a single surface frontier is

$$f(r) = \begin{cases} 0, & \text{if } r < d_s \\ t_m, & \text{if } d_s \leq r < d_m \\ \dfrac{A_c N}{A_s}, & \text{if } d_m \leq r \end{cases} \tag{2}$$

Where $d_m = \sqrt{\frac{h^2 N}{4\pi t_m}}$, $A_c = h^2$ is the area of one face of the cell and $A_s = 4\pi r^2$ is the area of a sphere. Equation 2 is plotted in Figure 4 using our vehicle's sensor parameters and the thresholds.

For a given view we can evaluate Equation 2 for each unocculded surface frontier and sum the results to give us a view utility. We can repeat this over a set of views to create a 3D utility function. To demonstrate this utility function we evaluate Equation 2 densely in Figure 5 without thresholds (i.e. $t_m = \infty$ and $d_s = 0$).

Unfortunately, it is expensive to evaluate the utility function densely over the map because of the ray tracing required. From Figure 5 we observe that the surface frontier observation utility decreases as distance from the surface increases. If our goal is to maximize a view's utility, the view can be offset from the surface between the safety distance d_s and d_m.

5.3 View Planning

Our view planning approach offsets the occupied cells using a distance transform then uniformly samples potential views on the offset surface. The interested reader is invited to read more about the SPARTAN path planner [4] which details our distance transform and view sampling approach. For each view found using SPARTAN, we determine which surface frontiers are visible by ray tracing. For a given view we sum Equation 2 evaluated for each visible surface frontier cell. This gives us a set of views weighted by their utility as shown in Figure 6. Given a robot position, partial map, and exploration views, we plan to a views using SPARTAN.

If we want to consider the utility of a proposed path we could incrementally sum the utility of views along a path, updating the map after summing each view by simulating frontier observations. In particular, an approach such as [14] could be

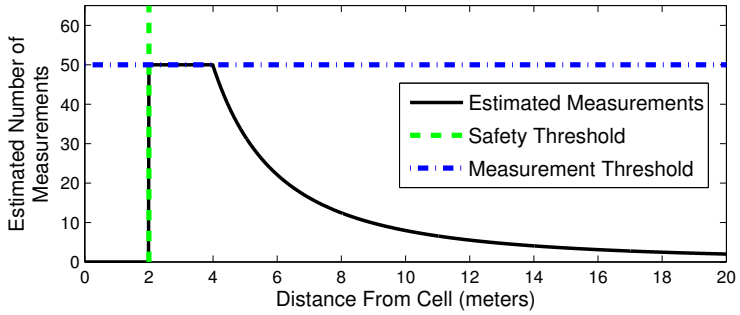

Fig. 4 Equation 2 evaluated with $h = 0.5$ meters, $N = 40000$, $d_s = 2$ meters, and $t_m = 50$

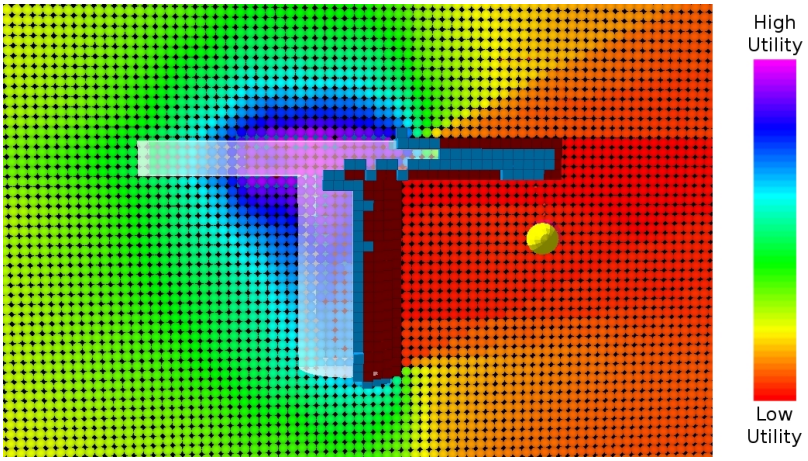

Fig. 5 Equation 2 evaluated during a simulated exploration over the center line of a simple bridge-like object. Red cells are occupied and blue cells are surface frontiers. The yellow sphere is a goal that the robot has almost reached.

implemented. Unfortunately, such solutions are computationally expensive and may quickly change due to new (real) observations. Instead of explicitly computing the utility of paths, we focus on using path cost to trade off between nearby exploration goals and distant exploration goals. We can then use Equation 3 to determine the highest value exploration view. For a given map, the exploration value V of a view X_f is

$$V(X_f, X_o) = \frac{U(X_f)}{U_{max}}\alpha - \frac{C(X_f, X_o)}{C_{max}}(1-\alpha) \tag{3}$$

Where α is a tuning parameter, $U(X_f)$ is a view utility, and $C(X_f, X_o)$ is the path cost for navigating from the current state X_o to view X_f. We can trade off between path cost and view utility by tuning $0 \leq \alpha \leq 1$. In large environments setting $\alpha = 1$ leads the MAV to inefficiently travel back and fourth across free space as it chooses maximum utility views. Reducing α increases the value of local view utility leading the robot to explore a region before travelling long distances to high utility views.

Once a view is chosen, the MAV replans paths at a fixed frequency until a) the view is reached, b) the view utility is reduced to zero, c) the view is deleted when the distance transform is updated with new observations, or d) the MAV cannot reach the view after a reasonable amount of time. When one of these conditions is met, a new view is chosen. If all samples are close to zero, the exploration planner terminates. There may be surface frontiers left when the algorithm terminates, but they will not be observable from reachable views.

To begin exploration the MAV could search the bounding box until a surface is detected to begin exploration. In our current implementation we assume that the

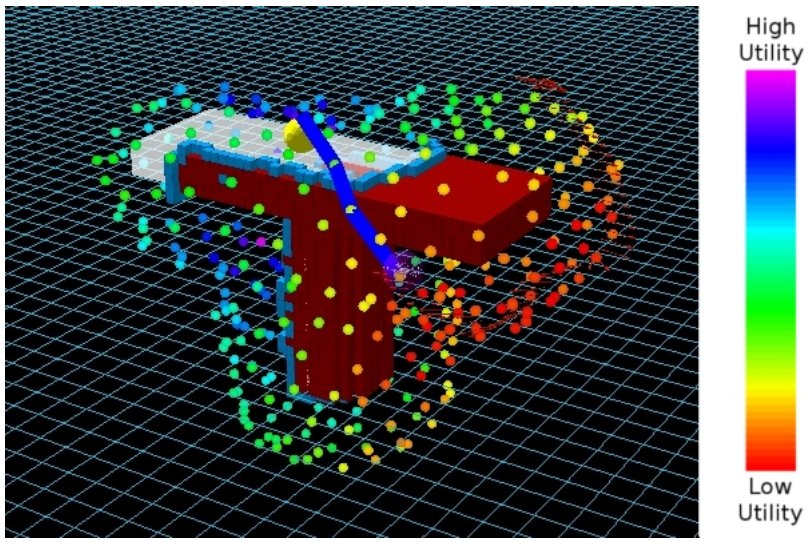

Fig. 6 Weighted views during exploration simulation. Red cells are occupied and blue cells are surface frontiers. The yellow sphere is a goal that the robot is navigating towards.

MAV starts with the structures surface in the sensor field of view and we assume the structures surface in the bounding box $S_k^* \cap R$ is one connected set. This speeds up data collection by making searching the boundary unnecessary.

6 MAV Platform

The vehicle is built on top of an oct-rotor platform shown in Figure 7. All processes are run on an onboard flight computer using a Intel i7 dual core 2.5GHz processor. Our mapping, planning, and controls processes communicate using the Robot operating system [10]. The flight computer sends yaw, pitch, roll, and thrust commands to a Mikrokopter flight controller. A cascaded PID controller is used to follow paths at a fixed 0.5 m/s. The vehicle weighs 5kg and has a flight time of approximately 15 minutes.

The range sensor used for mapping is a Hokuyo UTM-30LX-EW scanning lidar with custom gimbal which sweeps the laser in a spherical pattern. All lidar data is stored for point cloud generation, but only measurements within 15 meters are added to the occupancy grid. Upward and downward facing IDS imaging UI-1241LE-C-HQ cameras using Sunex DSL215B 185° fisheye lenses give the vehicle a spherical field of view. The cameras are downsampled to 480 X 480 pixels. The vehicle has a barometric pressure sensor and Microstrain 3DM-GX3-35 IMU reporting readings at 20Hz and 50Hz respectively. All sensors are time synchronized using a time server microcontroller. Depth enhanced Visual odometry [17] is run online at 30Hz. An unscented kalman filter [3] is used to fuse IMU, Visual odometry, barometric pressure, and GPS.

Fig. 7 MAV platform

7 Results

We validate our algorithm by autonomously modeling a train bridge in Pittsburgh, Pennsylvania. The bridge, shown in Figure 9, is a 50 meter long steel and concrete structure. The environment is vegetated in some places and confined by bridge structural members in others. Although there was little wind during the tests the environment is challenging for a MAV with intermittent GPS signal as the vehicle only has a two meter margin in some areas between its position and the obstacles.

At the start of the trial, the MAV is initialized with the bridge in sensor range. The bounding box parameters are loosely defined around the bridge and sent to the MAV. For this trial, the MAV flies to next best views without considering path cost (i.e. $\alpha = 1$). After manual takeoff the MAV is switched into autonomous mode and begins exploration. Since all processes run on board, communication between the MAV and a ground station is not used. The autonomous run lasted six minutes, and resulted in a five million point model of the bridge which is shown in Figure 1.

We compare autonomous exploration against a manual flight by skilled pilot. The pilot was instructed to scan all surfaces of the bridge using only a remote control to guide the vehicle. Figure 8 shows surface information vs time for the autonomous and manual trials. The results show that the autonomous exploration planner varies in performance by ± 8000 observed occupied cells when compared to the performance of the single flight by a skilled pilot.

Qualitatively, the MAV's behavior during the autonomous trials was similar to manually guided trial. During the autonomous trials the MAV maintained safe distance from bridge surface and obstacles like tree branches and tall grass. Figure 9 shows that the MAV chose a exploration path that varied from the human operator's path, spending most of the mission flying along the sides of the bridge instead of systematically weaving under the bridge like the operator.

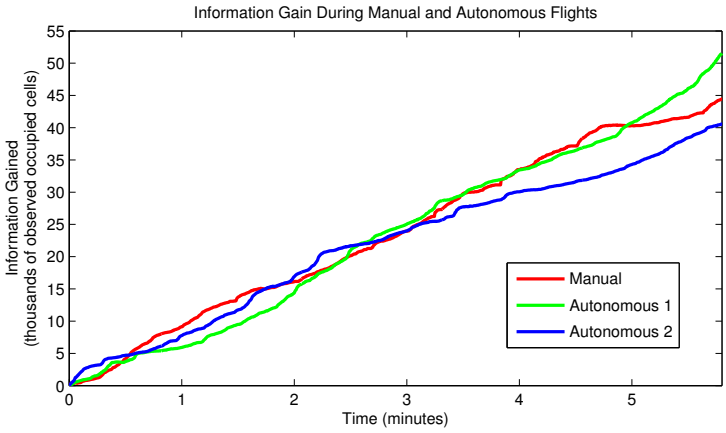

Fig. 8 Autonomous and manual flight surface information gain

8 Conclusion and Future Work

In this work we demonstrate an exploration planning algorithm that requires simple operator input to generate complex exploration paths for building a 3D model of an arbitrary structure outdoors. We show that a prior map is not necessary for planning paths for such a system. Finally, we demonstrate that autonomous vehicles using our exploration algorithm can perform as well as a skilled pilot.

There are a number of limitations in our method that present challenges for future work. The large 0.5 meter occupancy grid cell size used in this work limit the MAV's ability to detect small surface frontiers. If we can significantly decrease cell size using a map representation like an octree we could ensure coverage up to a resolution defined by the user. This would allow the user to trade off between point cloud resolution and flight time. In larger scale environments the travel cost between view points will become significantly higher making the consideration of path cost more important. Our system already supports trading off between view value vs path cost, but thorough analysis is needed to justify this approach in larger scale environments. Finally, this work assumes that the vehicle's position estimate has low uncertainty. This is a reasonable assumption considering the environment used in this paper and the required map accuracy. A higher mapping accuracy would make the system useful in more applications. To do this we would like to employ active localization techniques as well as improved sensors in future work.

References

1. Benjamin Adler, Junhao Xiao, and Jianwei Zhang. Autonomous exploration of urban environments using unmanned aerial vehicles. *Journal of Field Robotics*, 31(6):912–939, 2014.

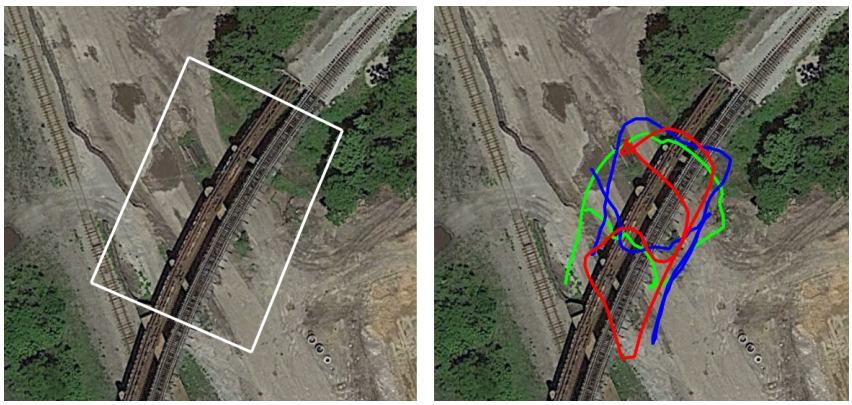

Fig. 9 MAV path and point cloud built during autonomous flight (blue) and manual flight (red)

2. P.S. Blaer and P.K. Allen. Data acquisition and view planning for 3-d modeling tasks. In *Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on*, pages 417–422, Oct 2007.
3. Andrew D Chambers, Sebastian Scherer, Luke Yoder, Sezal Jain, Stephen T. Nuske, and Sanjiv Singh. Robust multi-sensor fusion for micro aerial vehicle navigation in gps-degraded/denied environments. In *In Proceedings of American Control Conference, Portland, OR*, June 2014.
4. Hugh Cover, Sanjiban Choudhury, Sebastian Scherer, and Sanjiv Singh. Sparse tangential network (spartan): Motion planning for micro aerial vehicles. In *International Conference on Robotics and Automation*, May 2013.
5. C. Dornhege and A. Kleiner. A frontier-void-based approach for autonomous exploration in 3d. In *Safety, Security, and Rescue Robotics (SSRR), 2011 IEEE International Symposium on*, pages 351–356, 2011.
6. Lionel Heng, Dominik Honegger, Gim Hee Lee, Lorenz Meier, Petri Tanskanen, Friedrich Fraundorfer, and Marc Pollefeys. Autonomous visual mapping and exploration with a micro aerial vehicle. *Journal of Field Robotics*, 31(4):654–675, 2014.
7. Geoffrey A Hollinger, Brendan Englot, Franz S Hover, Urbashi Mitra, and Gaurav S Sukhatme. Active planning for underwater inspection and the benefit of adaptivity. *The International Journal of Robotics Research*, 32(1):3–18, 2013.
8. Sezal Jain, Stephen T. Nuske, Andrew D Chambers, Luke Yoder, Hugh Cover, Lyle J. Chamberlain, Sebastian Scherer, and Sanjiv Singh. Autonomous river exploration. In *Field and Service Robotics, Brisbane*, December 2013.
9. Bradley D. Null and Eric D. Sinzinger. Next best view algorithms for interior and exterior model acquisition. In *Proceedings of the Second International Conference on Advances in Visual Computing - Volume Part II*, ISVC'06, pages 668–677, Berlin, Heidelberg, 2006. Springer-Verlag.
10. Morgan Quigley, Ken Conley, Brian P. Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, and Andrew Y. Ng. Ros: an open-source robot operating system. In *ICRA Workshop on Open Source Software*, 2009.
11. William R. Scott, Gerhard Roth, and Jean-François Rivest. View planning for automated three-dimensional object reconstruction and inspection. *ACM Comput. Surv.*, 35(1):64–96, March 2003.
12. R. Shade and P. Newman. Choosing where to go: Complete 3d exploration with stereo. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 2806–2811, May 2011.
13. Shaojie Shen, Nathan Michael, and V. Kumar. Autonomous indoor 3d exploration with a micro-aerial vehicle. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 9–15, May 2012.
14. Amarjeet Singh, Andreas Krause, and William J. Kaiser. Nonmyopic adaptive informative path planning for multiple robots. In *Proceedings of the 21st International Jont Conference on Artifical Intelligence*, IJCAI'09, pages 1843–1850, San Francisco, CA, USA, 2009. Morgan Kaufmann Publishers Inc.
15. C. Stachniss, G. Grisetti, and W. Burgard. Information gain-based exploration using rao-blackwellized particle filters. In *Proc. of Robotics: Science and Systems (RSS)*, Cambridge, MA, USA, 2005.
16. B. Yamauchi. A frontier-based approach for autonomous exploration. In *Computational Intelligence in Robotics and Automation, 1997. CIRA'97., Proceedings., 1997 IEEE International Symposium on*, pages 146–151, 1997.
17. Ji Zhang, Michael Kaess, and Sanjiv Singh. Real-time depth enhanced monocular odometry. In *Intelligent Robots and Systems (IROS), Chicago, IL, USA*, September 2014.